RULE DISCOVERY IN DATABASES

By

HYONTAI SUG

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

1998

To my Parents and Family

ACKNOWLEDGEMENTS

I thank all the people who may be related to this research directly and indirectly. Special gratitute should be given to my mentor, Dr. Douglas D. Dankel II, who showed me great understanding and love for students, and my supervisory committee members, Dr. Paul W. Chun, Dr. L.M. Fu, Dr. S. Chakravarthy, and Dr. H. Beck, who have not hesitated to give me valuable lessons for the right direction of the research, and also to colleague graduate students who have studied the data mining area together, J. Kim, Y. Shinozaki, and S. Winterstein.

The most appreciation and apology from me should be given to my wife, Aee-Ran Eum, who has not hesitated to encourage me at the time of difficulty, and my two children, Choon-me and Joo-me, who have endured my absences for a long period of time until the completion of the work without much understanding of why they should, missing her and their beloved at the other side of the globe.

TABLE OF CONTENTS

Abstract of Dissertation Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy

RULE DISCOVERY IN DATABASES

By

HYONTAI SUG

December, 1998

Chairman: Dr. Douglas D. Dankel II
Major Department: Computer and Information Science and Engineering

Several problems exist for data mining in real world databases: the difficulty of determining a decision attribute when limited domain knowledge exists, or the difficulty in selecting a decision attribute from a new table formed from joining several relations, the problem of reflecting users' interests in the knowledge discovery process, the problem of finding a better prediction model for very large databases, and the difficulty in understanding large decision trees generated from very large target databases. This dissertation presents research results to solve these problems using methods that, first, determine a good decision attribute based on an approach developed from rough set theory and decision tree generation; second, discover optimal descriptive rule sets with respect to users' interests in stepwise refinement manner to cope with high dimensional and voluminous data sets which may contain numerous manifest facts; third, provide a suboptimal prediction model called HDT (hybrid decision tree) for very large databases. Finally, this dissertation presents a method to

find attribute sets based on frequent patterns in a joined table of relational databases which are used to generate more understandable and meaningful decision trees.

CHAPTER 1
INTRODUCTION

1.1 The Overview of Knowledge Discovery in Databases

As the use of computers has become wide spread, data are being collected at a dramatic rate across a wide variety of fields. As a result, there is a growing need to extract hidden information from the accumulated data.

Traditional methods of data analysis, which are mainly based on humans' dealing directly with data, do not scale well on the voluminous data. When the size and relationship between data are within human perception, an individual may devise hypotheses on the data set and prove these hypotheses by using statistical data analysis techniques. Today, with the dramatic rate of increase of data, the relationships between data are becoming more and more difficult for humans to perceive. To fulfill this need *data mining* and *knowledge discovery in databases(KDD)* have attracted a significant amount of attention [15, 21, 22, 23, 24, 35]. A general overview of processes of knowledge discovery in databases is shown in Figure 1.1. To cope with voluminous data most KDD systems focus on a segment of the database or perform

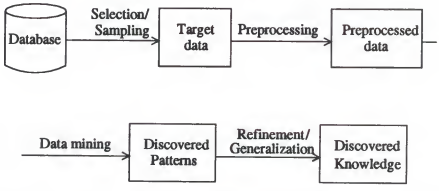


Figure 1.1. A General Overview of Processes of Knowledge Discovery in Databases

sampling to create a representative subset. After selecting a manageable amount of data, preprocessing is performed to convert any numeric values into symbolic values. Data mining algorithms are applied to the preprocessed data to find patterns under acceptable computational efficiency. Then refinement is performed by applying an interest measure on the discovered patterns to help humans ignore worthless patterns with respect to validity, novelty, and usefulness. Generalization is also performed to find a more general form of the found rules.

1.2 Data Mining Tasks

Two different goals exist for data mining in a practical sense: prediction and description [22]. Prediction focuses on foretelling unknown cases based on current data, while description focuses on finding human-interpretable patterns characterizing the data. For example, most stock price forecasting systems based on neural networks focus on prediction. Finding association rules, for example, *80% of customers who buy instant coffee also buy non-dairy creamer*, is an example of description.

To achieve these two goals we have the following principal data mining tasks [22]:

1. Classification maps a datum into one of the predefined classes.
2. Clustering is needed if the user does not define classes. So, systems first must discover subsets of related data items, then they have to find an appropriate description for each subset.
3. Summarization creates compact descriptions of subsets of data.

1.3 Solution Approaches

Because of the interdisciplinary nature of KDD, there are a variety of solution approaches [24]. Most common data mining technologies used today can be categorized as

- Neural Networks: since the translation of multi-layer neural networks into a symbolic form is still unknown, most neural network-based systems focus on prediction [29].

- Genetic Algorithms: the learned rule set of most genetic algorithm-based learning systems are for single class learning only. In other words, they can learn a rule set for positive members of a class, the examples not covered by the rule set are negative examples. So their applicable domains are limited [3, 16, 41, 48].

- Decision Trees: as the most successful and common data mining approach, decision tree-based systems have been used widely. There are two basic approaches: one is based on statistical tests applied to select the root attribute of a subtree like CART [5], and the other is based on the entropy measure of information theory like ID3 [62] and C4.5 [63].

- Other approaches include rule induction methods by adapting traditional inductive learning algorithms [59] like AQ15 [50], Bayesian networks [8, 9, 33, 64], Nearest Neighbor [12], regressions [5], association rules [2], and Rough sets [44, 53, 60, 73].

1.4 Motivation

1.4.1 How Can We Choose A Decision Attribute With Limited Domain Knowledge?

Since most KDD systems assume a good training data set has been selected during the data selection process, they usually assume that a decision attribute is predefined, that is, domain experts always can choose an appropriate field as a decision attribute. But in real world databases we may have some difficulty in determining a decision attribute if we have limited domain knowledge about the database. Moreover, if several relations are joined together, we may have a similar problem when we want to select a decision attribute from a new table [7, 18, 46, 77]. Even if we have a

L	Q	P
l_1	B	H
l_1	A	H
l_1	A	H
l_2	A	M
l_2	B	M
l_2	B	H

Figure 1.2. A Relation

single table structure, sometimes we might want to know the best decision attribute among several possibilities with respect to the confidence and complexity levels of the candidate rule set. Consider the relation in Figure 1.2 in which the attributes are the supplier's location (L), the quality of supplied goods (Q), and the price of supplied goods (P). Attribute Q can have value A or B, and attribute P can have value medium (M) or high (H). We may pose the question:

Which is better decision attribute, quality or price?

When price is the decision attribute, we have the possible rules:

if (supplier's location = l_1) then (price = high) 100%, and
if (supplier's location = l_2) then (price = medium) 67%.

When quality is the decision attribute, we have the possible rules:

if (supplier's location = l_1) then (quality = A) 67%, and
if (supplier's location = l_2) then (quality = B) 67%.

When price is the decision attribute, the rules have a higher confidence. In addition to the confidence there is the possibility that we may have rule sets of different complexity depending on the selected decision attribute.

Since the table is very simple, we can determine the rules and compute related confidences. But when the database contains a significant amount of data, manual computation is impossible.

1.4.2 How Can A Data Mining System Reflect The Users' Interests?

Although most KDD systems are based on machine learning approaches, it is almost impossible to directly apply some existing machine learning techniques because most target databases for knowledge discovery are very large and contain numerous manifest facts. So, we need some selection or sampling process [56] to select items from the target database. But the selection process performed by humans is very time consuming due to its ill-defined nature [22] and the found knowledge based on samples is prone to sampling errors. An alternative and better strategy may be to use the raw, intact database with some bias on the discovered knowledge. But, exhaustive rule set generation may not be a good idea since it is computationally very expensive and a large portion of the discovered knowledge may be manifest facts. Since the purpose of most KDD systems is to find hidden information that governs some significant part of population, the discovery of rare cases is pointless. If we take advantage of the fact that more general concepts occur more frequently, stepwise refinement of the discovered knowledge is a more appropriate strategy.

1.4.3 How Good Are Conventional Decision Trees In Prediction?

When we build decision trees, the root node of each subtree is chosen among the attributes which have not yet been chosen by ancestor nodes so that the selected attribute is the best split based on the average goodness of the split in a broad sense. Decision trees built in this way result in average to good prediction models for the target data. But, a single tree may lead to many unnecessary tests of attributes and may not represent rules that are best for some substantial subset or collection of the

objects in the database. In other words, these trees only can find a good prediction on average cases. Depending on the application, we may need very accurate prediction rules which check only a few critical feature values. Otherwise the classes of the objects may be decided more accurately by using additional rules of high confidence that are not in the decision tree, and possibly without testing additional attribute values.

1.4.4 How Can We Make More Understandable and Meaningful Decision Trees?

The decision tree induction method has been developed in pattern recognition and machine learning field where usually the training data size is small. Recently, the decision tree generation for very large databases has been reported successful [6, 49, 78] neglecting the fact that the generated decision tree may have understandability problem due to their size. Because the target databases of KDD are usually very large and contain many manifest facts, the direct application of decision tree generation methods may create a large decision tree of which most of branches may be meaningless. The conventional way to solve this problem is to rely on a careful data selection process before the data are supplied to the data mining step. The selection process relies heavily on human interaction with the system to select a meaningful set of data since it is not easy to impose background/domain knowledge that is as capable as a human's. Thus the data selection process is very time consuming. One way to solve this problem is to make the users participate in the decison tree generation process so that the generated tree can be more meaningful. To make the user interaction a feasible task, it is very important to generate smaller decision trees—intermediate trees or subtrees—to aid user comprehension, so some guide is needed to select attribute sets that lead to smaller trees with lower error rates.

1.5 Contribution To Knowledge

As discussed in the motivation, this research has the goal to devise algorithms to solve the problems of choosing a good decision attribute, finding an optimal rule set with respect to users' interests, supplementing the weak points of existing decision trees in predictional power for KDD application, and finally generating more understandable/meaningful decision trees.

Our research developed methods to surmount the limitations of the current systems as related above. The contribution of our research is the development of

- A novel decision attribute finding method that is based on the dependency of each attribute using rough set theory and decision tree complexity based on an entropy measure to determine a good decision attribute.
- An optimal descriptive rule set finding method for large database tables with respect to a user's interest which copes with high dimensional and voluminous data sets that may contain numerous manifest facts.
- A hybrid decision tree generation method that can provide not only accurate predictions but also diminished tests on attribute values.
- A new approach to generate more understandable/meaningful decision trees with respect to size, shape, and users' interests from frequent patterns/rules that may exist in a joined table of relational databases.

1.6 Outline

Chapter 2 discusses related work, and Chapter 3 discusses the background of this research, rough set theory, association rules, and decision trees. Our developed methods are presented in chapters 4, 5, 6, and 7. Chapter 4 presents a new method to determine a good decision attribute among several possibilities. Chapter 5 presents

our stepwise rule set discovery method that can allow users to find an optimal descriptive rule set with respect to their interests. Chapter 6 presents our method to build a HDT (hybrid decision tree) that broadens the applicability of conventional decision trees. Chapter 7 presents a method to find more meaningful decision trees that can cope with the voluminous nature of real world databases, and finally conclusions and future research are presented in Chapter 8.

CHAPTER 2
RELATED WORK

KDD research including machine learning can be divided into two categories, supervised and unsupervised learning. Classification belongs to supervised learning, whereas clustering belongs to unsupervised learning.

Many classification systems have been implemented including decision tree systems [5, 50, 63], neural networks [29], and nearest neighbor method like lazy decision trees [28], rough set-based systems [19, 36, 38, 66], and so on. The basic assumption in all of these approaches is that we know the class of each example beforehand. Most rough set theory-based systems including Hong and Mao [36], Hu [38], and Han and Hu [32] assume that a decision attribute is given by the user and use a taxonomy supplied by the user to generalize lower to higher level rules. Decision trees including lazy decision trees and rough set theory are discussed in detail in Chapter 3.

Although most classification systems assume that a decision attribute is given, this assumption can be a limitation for the applicability of these systems since it is not always clear that we know exactly which attribute in a database table is the decision attribute. In other words, data mining systems based on classification are a legacy of machine learning systems which rely on teachers to give good training examples. Due to this, an elaborate selection process is needed unless the database has been arranged in a very simple structure.

When we do not know the exact class of each object or row of a database table, we should rely on clustering. Earlier work in this area has been mostly performed outside artificial intelligence under the name of cluster analysis using numerical taxonomy

and distance measures to define similarity between objects. Much work in pattern recognition is based on this method [17].

The weak points of this method are as follows:

- The thresholds separating one cluster from another are prone to be arbitrary.
- There is limited use of semantic information about the objects.

Conceptual clustering methods have been developed to take advantage of the object's semantics. CLUSTER/S [40] uses background knowledge to determine the class of an object. But there is no guarantee that the built–in background knowledge can be very helpful for near-optimal clustering, and evaluation using LEF(lexical evaluation function) is prone to be arbitrary, and difficult to justify.

The HUATUO system [10] was made for traditional Chinese medicine. The system used a distance measure which counts how many different items exist between each rules. Fortunately, the taxonomy generated by the system matched the real world taxonomy. The problem with this approach is there is no guarantee that the used distance measure can be applied to other domains.

COBWEB [27] defines classes as a probability distribution over the values of attributes of objects and generates a hierarchy of classes. The system uses a category utility measure which is similar to the Gini index [5]. AUTOCLASS [9] creates a similar hierarchy of clusters based on the Bayesian classification. The tree generated by COBWEB has the property that all the nodes except the root node define a class. If we have the object table shown in Figure 2.1, COBWEB generates the probabilistic categorization tree in Figure 2.2 (adapted from Fisher [27]). The number by each attribute value is the probability of this item. Note that in Figure 2.2 the root comprises all objects, while the left subtree (class 1) comprises objects 1, 3, 5, and 7, and the right subtree (class 2) comprises objects 2, 4, 6, and 8. Class 3 contains objects 1 and 3, and so forth.

Obj.#	Size	Shape	Color
1	small	square	blue
2	medium	sphere	red
3	small	square	blue
4	medium	sphere	red
5	small	square	grey
6	large	sphere	red
7	small	square	grey
8	large	sphere	red

Figure 2.1. An Object Table

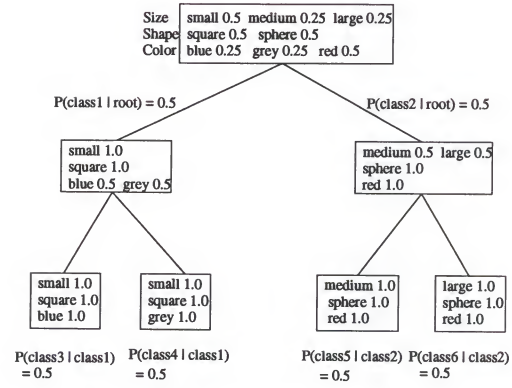


Figure 2.2. A Probabilistic Categorization Tree

COBWEB generates a taxonomy based on the probability distribution. But this taxonomy may differ from a human's since no external knowledge is used for the classification, so that it must rely on the human's verification of the generated tree. Another weak point is its categorization (note that all nodes are class definitions). Unless the sample used to generate the tree is as good as the population, the generated tree does not correctly reflect the statistical distribution of the population. Therefore, to generate a statistically valid taxonomy, the database should have a good set of data so that it properly reflects the population. So, incremental learnability is important in COBWEB [25, 26].

Another related family of KDD systems are the association rule discovery systems [2]. In a relatively short period of time since the first appearance of the algorithm in 1993 [1], much research has been done using this approach. In these systems, researchers try to find frequent patterns from transactional databases, like "80% of customers who buy instant coffee also buy non–dairy creamer." Association rule discovery systems and their limitations are discussed in Chapter 3 since they are one of the major ingredients of this research.

ARCS (association rule clustering system) [45] uses an association rule clustering algorithm when the righthand side of the rules contains a single fixed value and the lefthand side of the rules consists of two ordered attribute values. For example, If (age = x) AND (salary = y) Then NUGGET.

ARCS uses an image processing technique known as a low-pass filter to smooth the two dimensional grid as a method of generalization. Unfortunately, since categorical attributes have no ordering, ARCS has to consider all the combination of attribute values to make a grid table. Two problems exist: cases involving more than two attributes become exponentially complicated, and multiple righthand side values bring about a problem with the rule representation.

In a genetic algorithm-based machine learning system like GABIL [16], the correctness of a population, that is, the rule set being trained, to the training examples plays a very important role in the evolution process. Therefore, we can think of the system as a pattern finding system which also relies on randomized search. The weak points of most genetic algorithm-based learning systems are their computational intensiveness if the size of the training examples is large, and their limit to allowing only positive/negative classes [3, 41, 48].

In this chapter we discussed the related work in KDD systems and machine learning systems. In next chapter we discuss the background of our research, rough set theory, association rules, and decision trees.

CHAPTER 3
BACKGROUND

Rough set theory, the association rule discovery method, and the decision trees are three basic ingredients of our research. This chapter presents a summary of each.

3.1 Rough Set Theory

Rough set theory is a mathematical tool to deal with vagueness and uncertainty of imprecise data. After being introduced by Z. Pawlak in the early 1980s, the theory has been developed and expanded to include applications in the fields of decision analysis, data analysis, pattern recognition, machine learning, and knowledge discovery in databases [24, 60, 66].

Rough set theory overlaps with the Dempster-Shafer theory of evidence in dealing with vagueness and uncertainty [79], but these methods have different focuses. While the Dempster-Shafer theory of evidence focuses on calculation of belief and plausibility values [14, 30], rough set theory focuses on making use of sets that have *lower and upper approximations.* Skowron and Grzymala-Buse [79] give an interpretation of the belief and plausibility functions in terms of the lower and upper approximation of sets. *The lower approximation of the concept consists of all objects which surely belong to the concept, whereas the upper approximation of the concept consists of all objects which possibly belong to the concept.* The main advantage of rough set theory is that concept approximation is *solely based on data*, so it does not need any preliminary or additional information about the data.

3.1.1 The Principal Concept of Rough Sets

Knowledge base

If we are given a finite set U $\neq \emptyset$ of objects, called a universe, and a family of equivalence relations over U, called **R**, then a relational system K=(U,**R**) is a knowledge base.

An equivalence relation represents the set of values that each object can have as one of its properties.

- If R is an equivalence relation over U then U/R means the family of all equivalence classes of R (or the classification of U) referenced to as categories or concepts of R.
- $[x]_R$ means a category in R containing an element $x \in U$.
- IND(**P**) means the intersection of all equivalence relations belonging to **P** and is called an indiscernability relation over **P** where $\mathbf{P} \subseteq \mathbf{R}$ and $\mathbf{P} \neq \emptyset$.
- U/IND(**P**) means the family of all equivalence classes of the equivalence relation IND(**P**).

Example. Suppose we have a knowledge base of toy blocks K=(U,(color,shape)) where U=$\{x_1, x_2, x_3, x_4\}$ and we have two equivalence relations, color and shape. The set of toy blocks U can be classified according to color and shape as follows.

By color:

x_1, x_3 are red, and

x_2, x_4 are blue.

By shape:

x_1, x_2, x_3 are round, and

x_4 is square.

That is, U/color = { { x_1, x_3 }, { x_2, x_4 } }, and U/shape = { { x_1, x_2, x_3 }, { x_4 } }.

Thus,

$[x_1]_{color}$ = red, $[x_1]_{shape}$ = round,

$[x_2]_{color}$ = blue, $[x_2]_{shape}$ = round,

$[x_3]_{color}$ = red, $[x_3]_{shape}$ = round, and

$[x_4]_{color}$ = blue, $[x_4]_{shape}$ = square.

IND({ color, shape }) identifies objects that have the same values of color and shape. So, U/IND({ color, shape }) = { { x_1, x_3 }, { x_2 }, { x_4 } }. Since x_1 and x_3 have the same values of color and shape, they are an equivalence class.

Rough sets

Rough sets are sets that are defined using two approximations, *upper and lower approximations.*

Let X $\subseteq$ U and R $\in$ IND(R) then

- R-lower approximation, $\underline{R}$X = { Y $\in$ U/R: Y $\subseteq$ X },
- R-upper approximation, $\overline{R}$X = { Y $\in$ U/R: Y $\cap$ X $\neq \emptyset$ }, and
- R-boundary region of X, BN_R(X) = $\overline{R}$X - $\underline{R}$X.

Additionally, we can define the following terms:

- R-positive region of X, POS_R(X) = $\underline{R}$X,

- R-negative region of X, $NEG_R(X)$ = U - $\overline{R}X$, and
- R-borderline region of X, $BN_R(X)$.

Proposition.

- X is R-definable iff $\underline{R}X = \overline{R}X$.
- X is rough with respect to R iff $\underline{R}X \neq \overline{R}X$.

The lower and upper approximation can also be represented in the following equivalent forms:

- $\underline{R}X = \{ x \in U: [x]_R \subseteq X \}$, and
- $\overline{R}X = \{ x \in U: [x]_R \cap X \neq \emptyset \}$.

Example. Suppose we have the following equivalence classes:

U/color = $\{ \{ x_1, x_2, x_3 \}, \{ x_4, x_5, x_6, x_7, x_8 \} \}$, and

U/shape = $\{ \{ x_1, x_4 \}, \{ x_2, x_5, x_7 \}, \{ x_3, x_6, x_8 \} \}$. Assume R = { color, shape }.

Thus U/R = $\{ \{ x_1 \}, \{ x_2 \}, \{ x_3 \}, \{ x_4 \}, \{ x_5, x_7 \}, \{ x_6, x_8 \} \}$.

The objects are divided by decision D with the value { 0, 1 }. The set of objects with decision 0 is $D_0 = \{ x_1, x_4, x_5, x_8 \}$, and the set of objects with decision 1 is $D_1 = \{ x_2, x_3, x_6, x_7 \}$.

By the definition of R-lower approximation,

$$\underline{R}D_0 = \bigcup \{ Y \in U/R: Y \subseteq D_0 \} = \{ x_1 \} \cup \{ x_4 \} = \{ x_1, x_4 \}.$$

By the definition of R-upper approximation,

$$\overline{R}D_0 = \bigcup \{ Y \in U/R: Y \cap D_0 \neq \emptyset \} = \{ x_1 \} \cup \{ x_4 \} \cup \{ x_5, x_7 \} \cup \{ x_6, x_8 \}$$

$= \{ x_1, x_4, x_5, x_6, x_7, x_8 \}$.

R-boundary of D_0 $BN_R(D_0) = \overline{R}D_0 - \underline{R}D_0 =$

$\{ x_1, x_4, x_5, x_6, x_7, x_8 \} - \{ x_1, x_4 \} = \{ x_5, x_6, x_7, x_8 \}$.

R-negative region of D_0 $NEG_R(D_0) = U - \overline{R}D_0 =$

$\{ x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8 \} - \{ x_1, x_4, x_5, x_6, x_7, x_8 \} = \{ x_2, x_3 \}$.

Similarly,

$\underline{R}D_1 = \{ x_2, x_3 \}$,

$\overline{R}D_1 = \{ x_2, x_3, x_5, x_6, x_7, x_8 \}$, and

$BN_R(D_1) = \{ x_5, x_6, x_7, x_8 \}$.

Moreover,

$\underline{R}(D_0 \cup D_1) = \cup \{ Y \in U/R: (Y \subseteq D_0) \cup (Y \subseteq D_1) \}$

$= \{ x_1, x_4 \} \cup \{ x_2, x_3 \} = \{ x_1, x_4, x_2, x_3 \}$, and

$\overline{R}(D_0 \cup D_1) = \cup \{ Y \in U/R: (Y \cap D_0 \neq \emptyset) \cup (Y \cap D_1 \neq \emptyset) \}$

$= \{ x_1, x_4, x_5, x_6, x_7, x_8 \} \cup \{ x_2, x_3, x_5, x_6, x_7, x_8 \}$

$= \{ x_1, x_4, x_2, x_3, x_5, x_6, x_7, x_8 \}$.

R-boundary of $D_0 \cup D_1$ $BN_R(D_0 \cup D_1) = \overline{R}(D_0 \cup D_1) - \underline{R}(D_0 \cup D_1)$

$= \{x_5, x_6, x_7, x_8 \}$.

The accuracy measure of rough set X with respect to equivalence relation R is defined by

$$\alpha_R(X) = (\text{card } \underline{R}X) \ / \ (\text{card } \overline{R}X), \text{ where } X \neq \emptyset.$$

Knowledge representation system (KRS)

In rough set theory KRS is a pair $S = (U, A)$ where

1. U is a nonempty, finite set called the universe,

Obj.#	P	Q	R
1	1	0	1
2	1	1	0
3	0	2	1
4	1	2	1
5	2	0	2
6	2	1	1
7	0	2	0
8	2	2	0

Figure 3.1. A KRS (Knowledge Representation System)

2. A is a nonempty, finite set called the primitive attributes,
   and $\forall$ a, a $\in$ A, a: U $\rightarrow V_a$ where V_a is the value set of attribute a.

We can represent a KRS in a tabular form.

If two objects have the same values with respect to a subset of attributes B $\subseteq$ A, they form an indiscernability relation represented by

IND(B) = { (x,y) $\in U^2$: for every a $\in$ B, a(x) = a(y) }.

Note that IND(B) has the same notation as an equivalence relation, thus, it has the same meaning.

Example. A KRS S = (U, A) where U = { x_1, ..., x_8 } and A = { P, Q, R } is represented in tabular form as shown in Figure 3.1.

The correspondence between knowledge base (KB) K = (U, **R**) and KRS S = (U, A) assumes $a_R \in$ A. If R $\in$ **R** and U/R = { X_0, ..., X_k } then we have

a_R: U $\rightarrow V_{a_R}$ where V_{a_R} = { 1, ..., k } and a_R(x) = i iff x $\in X_i$ for i = 1,...,k.

Note the above definition applies to every equivalence relation in **R**.

Example. If we represent the table in Figure 3.1 in KB form, we have

KB = (U, **R**) where U = { x_1, ... , x_8 }, **R** = { P, Q, R },
U/P = { P_0 = { x_3, x_7}, P_1 = { x_1, x_2, x_4 } P_2 = { x_5, x_6, x_8 } },
U/Q = { Q_0 = { x_1, x_5}, Q_1 = { x_2, x_6 } Q_2 = { x_3, x_4, x_7, x_8 } }, and
U/R = { R_0 = { x_2, x_7, x_8}, R_1 = { x_1, x_3, x_4, x_6 }, R_2 = { x_5 } }.

Reduction of knowledge

We may question whether all the information in a given table is necessary to classify all the objects based on the value of decision attributes. As you see from the mappings KB and KRS, the concept of decision attributes corresponds to rough sets. Our goal is to find a minimal set of values which can determine the category based on the given information. The whole process is called reduction.

In the context of reduction of knowledge we have two fundamental concepts, a reduct and the core.

Definition. Let **R** be a family of equivalence relations and R $\in$ **R**.

If IND(**R**) = IND(**R** - {R}) then R is dispensable in **R**
else R is indispensable in **R**.
If $\forall$ R, R $\in$ **R** is indispensable then the family **R** is independent
else the family **R** is dependent.

Definition. Let **P** $\subseteq$ **R**. If **P** is independent and IND(**P**) = IND(**R**), then **P** is a reduct of **R**, RED(**R**).

Obj.	A	B	C
x_1	0	0	0
x_2	1	2	2
x_3	2	0	3
x_4	0	2	3
x_5	2	0	0
x_6	3	1	1
x_7	3	2	2
x_8	1	2	2

Figure 3.2. ABC table

Definition. The set of all indispensable relations in **R** is called the core of **R**, CORE(**R**).

Example. Suppose we have the ABC table shown in Figure 3.2. If we represent the table in KB form, we have **R** = { A, B, C }, and

U/A = { { x_1, x_4 }, { x_2, x_8 }, { x_3, x_5 }, { x_6, x_7 } },
U/B = { { x_1, x_3, x_5}, { x_6 }, { x_2, x_4, x_7, x_8 } },
U/C = { { x_1, x_5}, { x_6}, { x_2, x_7, x_8 }, { x_3, x_4 } }, and
U/**R** = { { x_1 }, {x_2, x_8 }, { x_3}, {x_4}, {x_5}, {x_6}, {x_7 } }.

Let us see the dispensability of each equivalence relations.

U/IND(**R** - {A}) = U/IND({B,C})
= { { x_1, x_5}, {x_6}, {x_2, x_7, x_8}, {x_3}, {x_4} } $\neq$ U/IND(**R**).

We can compute the above value by comparing each element of U/B and U/C, and make a new set using the common items. Or from the table, we can directly group objects which have the same values with respect to A and B. Thus, A is indispensable.

U/IND(**R** - {B}) = U/IND({A,C})
= { { x_1, x_5}, {x_6}, {x_2, x_8}, {x_3}, {x_4}, {x_7} } = U/IND(**R**), so, Q is dispensable.

U/IND($\mathbf{R}$ - {C}) = U/IND({A,B})

= { { x_1, x_5}, {x_2, x_8}, {x_3}, {x_4}, {x_6}, {x_7} } = U/IND($\mathbf{R}$), so, R is dispensable.

Since P was found to be indispensable, P is the core of $\mathbf{R}$.

If we consider {A,B} and {A,C}, each family of equivalence relations is independent. Since

U/IND({A,B} - {A}) = U/IND(B) $\neq$ U/IND({A,B}),

U/IND({A,B} - {B}) = U/IND(A) $\neq$ U/IND({A,B}),

U/IND({A,C} - {A}) = U/IND(B) $\neq$ U/IND({A,B}), and

U/IND({A,C} - {C}) = U/IND(A) $\neq$ U/IND({A,C}), moreover,

U/IND({A,B}) = U/IND({A,C}) = U/IND({$\mathbf{R}$}).

Therefore, {A,B} and {A,C} are the reducts of $\mathbf{R}$.

In the above computation we can observe that the core is always included in the knowledge base, there may be several reducts, and any reduct can uniquely identify each object. Note that equivalence relations correspond to attributes in KRS. The method to compute a minimal rule set using the rough set theory can be found in Pawlak's book [60].

3.2 Association Rules

Association rules are rules about how often a set of items occur together. These rules produce information on the patterns or regularities that exist in a database. For example, suppose we have collected a set of transactional data concerning sales in a supermarket for some period of time. We might find an association rule from the data stating "instant coffee $\Rightarrow$ non–dairy creamer (80%)," which means "80% of customers who buy instant coffee also buy non–dairy creamer." Such rules can be

useful for decision making on sales, for example, determining an appropriate layout for items in the store.

The following is a formal definition of the problem [2].

Let $I = \{i_1, i_2, ..., i_m\}$ be a set of items that are sold in the supermarket. Let T be a collection of transactions, where each transaction has an itemset $X \subseteq I$ that has been purchased at the same time by a customer. Each transaction has a unique identifier.

An association rule is an implication of the form $Y \Rightarrow Z$ where $Y \subset I$, $Z \subset I$, and $Y \cap Z \neq \emptyset$. The confidence C% of the association rule $Y \Rightarrow Z$ implies that among all the transactions which contain itemset Y, C% of them also contains itemset Z. For an itemset $X \subset I$,

Support(X) = s is the fraction of the transactions in T containing X.

So, the confidence of a rule $Y \Rightarrow Z$ is computed as the ratio

$$\text{support}(Y \cup Z)/\text{support}(Y).$$

Note that $Y \cup Z$ means items in a basket when we shop at the supermarket. If somebody buys one itemset Y, and the other guy buys one itemset Y and one itemset Z, then the confidence of the rule is 1/2. In practical applications the *support number* is also often used to represent how many times an itemset occurs in T. The itemsets that occur more frequently than some given support level are called *frequent itemsets.*

Discovering all frequent itemsets is a nontrivial problem if the collection of transactions is large. If the cardinality of the set of items is $|I|$, then the total number of possible itemsets is $2^{|I|}$. One approach is to have counters for each itemset of $2^{|I|}$. By scanning the transactional database once, we can find the support number of each itemset. But this method is unpractical since the cardinality of the set of items will be too large. A clever technique to solve this computational complexity problem is

Transaction ID	Items
1	1 3 4
2	2 3 5
3	1 2 3 5
4	2 5

Figure 3.3. A Transactional Database

to use the characteristics of itemsets—frequent itemsets containing "N" items are composed of smaller itemsets containing "N–1" items.

Suppose we have a transactional database like the one shown in Figure 3.3 (adapted from Agrawal et al. [2]) and we have a given support number of 2.

At iteration 1, 2, and 3 we have the frequent itemsets of size 1, 2, and 3 as shown in Table 3.1, Table 3.2, and Table 3.3 respectively.

Table 3.1. Iteration 1

frequent itemset	support_number
{1}	2
{2}	3
{3}	3
{5}	3

Table 3.2. Iteration 2

frequent itemset	support_number
{1,3}	2
{2,3}	2
{2,5}	3
{3,5}	2

Table 3.3. Iteration 3

frequent itemset	support_number
{2,3,5}	2

As shown in the example, the items that occur in the previous iteration as members of the itemsets are the only ones that can occur in the current iteration. In general, the kth iteration consists of the following steps [65]:

1. The set of candidate k-itemsets is generated by 1-extension of the frequent (k - 1)-itemsets generated in the previous iteration.

2. Supports for the candidate k-itemsets are generated by a single pass through the database.

3. The itemsets that do not have the given minimum support are discarded, and the remaining itemsets are designated frequent k-itemsets.

3.2.1 Association Rule Finding Algorithms

The first association rule finding algorithm, AIS, was developed in 1993 [1]. SETM was proposed to solve the problem when the transactional database is a relational database [37], and the case when the transactional records are in a general purpose database is also considered [34]. Algorithms Apriori and AprioriTid were introduced to achieve efficiency [2]. Parallel versions were also proposed to obtain results more quickly [11, 65]. Sampling was suggested to cope with very large transactional databases [74, 75]. Other research related to association rules includes the generalization of discovered association rules [31, 67] and extraction of specific/interesting rule sets from the discovered association rules [42, 47, 76].

Let us examine the algorithm Apriori, which is one of the most efficient algorithms, shown in in Figure 3.4. All the other algorithms are similar except for specializations resulting in more efficient execution.

```
L[1]= {frequent 1-itemsets}
For = ( k=2; L[k-1]= empty_set; k++) do
   begin
   C[k] = apriori-gen(L[k-1]) //find new candidates.
   Forall transactions t in T do
      begin
      C[t] = subset(C[k],t) //C[t]: candidates contained in t.
      Forall candidates c in C[t] do
         c.count++;
      end
      L[k] = {c in C[k] | c.count >= minsup}
   end
Answer = Union of L[k]
```

Figure 3.4. Algorithm Apriori

The **apriori–gen** function takes all frequent (k-1) itemsets $L[k-1]$ as input argument, and returns all possible frequent k itemsets that can be made from the $L[k-1]$. The two main steps of the function are the join step, shown in Figure 3.5, and the prune step, shown in Figure 3.6.

Because all size k-1 subset of a size k itemset must exist in size k-1 itemsets, we need two itemsets which are different in one item to make the candidate itemset of next iteration. Note that the existance of one smaller size itemsets does not guarantee the longer size itemset of the minimum support number. So, a database checking step is needed. The formal proof on which the algorithm generates all possible frequent itemsets can be found in Toivonen's Ph.D. thesis [74].

The **subset** function checks if the candidate frequent itemset is in the database and uses a hashtree for fast comparison.

JOIN step:

```
Select p.item_1, p.item_2, ...,p.item_{k-1},q.item_{k-1}
from L[k-1, p], L[k-1, q]
  //make k-itemset from 2 (k-1) itemsets which are different
      one item each other.//
where p.item_1 = q.item_1, ...,p.item_{k-2} = q.item_{k-2},
      p.item_{k-1} < q.item_{k-1}
Insert them into C[k]
```

Figure 3.5. The Join Step

PRUNE step:

```
Delete itemsets in C[k] such that any (k-1)-subset of them
is not in L[k-1].
```

Figure 3.6. The Prune Step

3.2.2 Rule Generation

There are many potential rules for found frequent itemsets. Since we have all smaller size itemsets in a database, there can be (2^n-2) rules if n is the size of a frequent itemset. For example, suppose we have a frequent itemset ABC. From this itemset we can generate six rules describing potential purchase implications, that is, A $\rightarrow$ BC, B $\rightarrow$ AC, C $\rightarrow$ AB, AB $\rightarrow$ C, AC $\rightarrow$ B, BC $\rightarrow$ A. In general, an itemset consisting of N items results in (2^n-2) rules, so it is possible to generate a large number of candidate rules. To determine which rules are legitimate involves trying each item in the frequent itemset as a decision part and selecting the rule which has the largest confidence. For example, suppose the support number of itemset ABC is 3, for AB is 4, AC is 5, and BC is 6. Then the confidence of the rule AB $\rightarrow$ C is $3/4 = 0.75$, AC $\rightarrow$ B is $3/5 = 0.6$, and BC $\rightarrow$ A is $3/6 = 0.5$. So, we select the association rule AB $\rightarrow$ C since its confidence is the highest.

An alternative method for rule generation is to drop all rules which have lower confidence values than some given user threshold.

3.2.3 The Generalization of Association Rules

A characteristic of the rule generation process is that each rule is found solely based on the frequency of the pattern from which a rule is generated. Therefore, these rules are found without considering any potential relationships with other rules. If a frequent pattern of length n is found, only the one with the highest confidence is selected among the $O(2^n)$ possible rules of the length n pattern. The rules are not simplified and many rules are generated since by using a longer pattern we can create many smaller patterns. For example, if we have a pattern ABCD, we may have AB, AC, AD, BC, BD, CD, ABC, ABD, and BCD as patterns. As a result, various methods for presenting interesting rules only and generalizing the found rules have been suggested. Selecting only interesting rules from a large collection of the found association rules relies on background/domain knowledge. For example, the user can explicitly specify what is interesting and what is not with templates [42].

Srikant and Agrawal [67] tried to generalize the rules which have a common decision part (the righthand side of a rule) by using a predefined taxonomy. Because the association rule finding algorithm was invented to find rules from a transactional database, which has the property of a simple unique structure containing attributes like product_ID, quantities_sold, there seems to be no need to consider the possibility of finding a common decision attribute that might be applicable to the database.

Toivonen et al. [76] tried to simplify rules by dropping a common part of the condition attribute values if the simplified rule still has a similar confidence value, for example, if we have two rules,

C++ Programming, Database Systems $\Rightarrow$ Computer Network with confidence = 0.9

Transaction ID	Items
1	6 2 3
2	0 2 3
3	4 2 3
4	1 2 3
5	5 1 3

Figure 3.7. A Transactional Database

C++ Programming, Database Systems, and Operating Systems
$\Rightarrow$ Computer Network with confidence = 0.9,

the second rule is pruned.

He also clustered rules based on a distance measure. The distance between rules is measured by using the count of the number of rows where the rules differ using the equation,

$$d(X \rightarrow Z, Y \rightarrow Z) = |XZ| + |YZ| - 2 \times |XYZ|$$

where $|XYZ|$ means the frequency of itemset XYZ.

Note that Y does not occur in the itemset XZ, and X also does not occur in the itemset YZ, but the frequency of pattern $|XYZ|$ is reflected in $|XZ|$ and $|YZ|$. By subtracting $|XYZ|$, we can find the number of different rows.

For example, suppose we have the transactions as shown in the transactional database of Figure 3.7. If we assume that we find rules on patterns of frequency 2, that is, the given support number is 2, we have rules

(rule 1) $2 \rightarrow 3$ with confidence 1.0 (derived from transaction 1 - 4), and
(rule 2) $1 \rightarrow 3$ with confidence 1.0 (derived from transaction 4 and 5).

The distance between rule 1 and rule 2 is

Transaction ID	Items
1	2 3 4
2	1 2 3
3	1 2 5
4	1 3 5

Figure 3.8. Another Transactional Database

$$d(2 \rightarrow 3, 1 \rightarrow 3) = |2,3| + |1,3| - 2 \times |1,2,3|$$
$$= 4 + 2 - 2 \times 1 = 4.$$

On the otherhand, suppose we have another transactional database shown in Figure 3.8.

We have rules

(rule 3) $2 \rightarrow 3$ with confidence 0.67,

(rule 4) $1 \rightarrow 2$ with confidence 0.67,

(rule 5) $1 \rightarrow 3$ with confidence 0.67, and

(rule 6) $1 \rightarrow 5$ with confidence 0.67.

The distance between rule 3 and rule 6 is

$$d(2 \rightarrow 3, 1 \rightarrow 5) = |2,3| + |1,5| - 2 \times |1,2,3,5|$$
$$= 2 + 2 - 0 = 4.$$

In this case the distance between rule 3 and 6 is equal to the previous case even though the two rules have no common items. So, depending on a given threshold, rule 3 and 6 may be clustered together since the distance measure does not rely on semantics.

3.3 Decision Trees

3.3.1 Decision Trees Based on An Entropy Measure

It has been known that making the smallest decision tree is a NP-complete problem, so entropy-based decision tree generation methods were invented by Quinlan [62, 63] to find the near smallest decision tree. Entropy is used to measure the amount of information or the uncertainty about the information.

Definition. Shannon defined entropy to be

$$\mathrm{H(P(X))} = -\sum_{x \in X} P(x) lg P(x), \forall x \in X,$$

where P(x) represents the probability distribution of x ∈ X. In this equation lg means log-base 2.

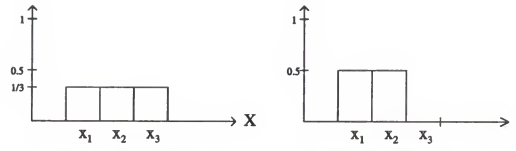
Example. Suppose we have a set X = {x_1, x_2, x_3} and the probability of each x_i is P(x_1)=1/3, P(x_2)=1/3, P(x_3)=1/3, then the uncertainty of the occurrence of these element is

$$\mathrm{H(P(X))} = -(\tfrac{1}{3}\lg\tfrac{1}{3} + \tfrac{1}{3}\lg\tfrac{1}{3} + \tfrac{1}{3}\lg\tfrac{1}{3}) = 1.6.$$

If we have a different probability distribution, P'(x_1)=1/2, P'(x_2)=1/2, P'(x_3)=0, then

$$\mathrm{H(P'(X))} = -(\tfrac{1}{2}\lg\tfrac{1}{2} + \tfrac{1}{2}\lg\tfrac{1}{2} + 0) = 1.$$

Comparing these two values, we know that the probability distribution of P(X) has larger uncertainty than P'(X). See the probability distribution of P and P' in Figure 3.9. Intuitively we can also state that P' has more certainty on which element could occur.



Probability distribution P Probability distribution P'

Figure 3.9. The Probability Distribution of P and P'

Example. Suppose we have two coins, one is fair and the other rigged, the probability of head and tail of the first is $\frac{1}{2}$, $\frac{1}{2}$, and probability of head and tail of the second is $\frac{9}{10}$, $\frac{1}{10}$. The entropy value or information content of the first one is

$$-(\tfrac{1}{2}\lg\tfrac{1}{2} + \tfrac{1}{2}\lg\tfrac{1}{2}) = 1,$$

while the entropy of the second one is

$$-(\tfrac{9}{10}\lg\tfrac{9}{10} + \tfrac{1}{10}\lg\tfrac{1}{10}) = 0.4691.$$

In other words, since the first coin has an equal chance for both heads and tails, we have no idea about which may occur making it a random choice. The first one has no information at all, but the second coin has good information since we can expect more heads. So, the principle is the smaller the entropy value, the more information we have.

Entropy also can be interpreted as the quantity of information conveyed by a message. Entropy depends on its probability and can be measured in bits which is computed as the negative of the logarithm to base two of the probability. For example, if we have sixteen equally probable messages, x_1, ..., x_{16}, the information conveyed by any one of them is $-\lg\frac{1}{16} = 4$ bits, which means that using only 4 bits we

	P_1	P_2	....	P_n
	a_1	a_2	....	a_n
class1	P_{11}	P_{21}	....	P_{n1}
.	.	.	....	.
.	.	.	....	.
.	.	.	....	.
.	.	.	....	.
class r	P_{1r}	P_{2r}	....	P_{nr}

Figure 3.10. Probabilities for the Values of an Attribute X

can distinguish the sixteen messages from each other. Since each message has equal probability, 1/16, the expected number of bits for the message is

$$\sum_{i=1}^{16} P(x_i)x_i = 16 \times (\tfrac{1}{16} \times (-\lg \tfrac{1}{16}) = 4 \text{ bits.}$$

Note that $x_i = -\lg P(x_i)$ for i=1, ..., 16, which is exactly the same equation as Shannon's entropy.

The computation in Quinlan's ID3 algorithm [62] is done as follows:

Suppose we have r classes each with probability of q_1, ..., q_r and we have an attribute X that can have n different values with probability of p_1, ..., p_n. The probability of each value for each class is p_{11}, ..., p_{1r}, p_{21}, ..., p_{2r}, ..., p_{n1}, ..., p_{nr}. See the table of probabilities for the values of an attribute X in Figure 3.10. The expected quantity of information for value a_i is

$$-\sum_{j=1}^{r} p_{ij} lg p_{ij},$$

so, the expected quantity of information for attribute X is

$$info_x(T) = \sum_{i=1}^{n} p_i(-\sum_{j=1}^{r} p_{ij} lg p_{ij})$$

where T is the probability variable of the whole population.

Since the above value has the property of "the less, the better," Quinlan used

$$\text{info(T)} = -\sum_{i=1}^{r} q_i lg q_i$$

as the minuend to reflect the certainty of the classification like the previous examples, and subtracted $info_x(T)$ from it. This Quinlan named the "gain,"

$$\text{gain(X)} = \text{info(T)} - info_x(T).$$

Since the "gain" criterion has a strong bias in favor of many-valued attributes, which fragment data more severely, Quinlan divided the "gain" with a term he called "split info" in C4.5 [63].

$$\text{split\_info(X)} = -\sum_{i=1}^{n} p_i lg p_i.$$

Note that p_i is the probability of each value of attribute X ignoring class membership. This value becomes large in proportion to the number of values in the attribute. The result is

$$\text{gain\_ratio(X)} = \text{gain(X)}/\text{split\_info(X)}.$$

Quinlan used frequency ratios instead of probabilities since computing the probabilities of all the probability variables is not easy. In this case,

p_i = (the number of a_i–valued objects) / (current total number of objects),

q_j = (the number of objects belonging to class j) /
(current total number of objects), and

p_{ij} = (the number of a_i–valued objects belonging to class j) /
(the number of objects belonging to class j in the current total number of objects).

Example. Suppose we have the data table in Figure 3.11. The table describes decisions with two condition attributes A and B.

Since we have nine positive cases and five negative cases

A	B	class
0	0	+
0	0	-
0	1	-
0	1	-
0	1	+
1	0	+
1	1	+
1	0	+
1	1	+
2	0	-
2	0	-
2	1	+
2	1	+
2	1	+

Figure 3.11. A Data Table

$$\text{info(T)} = -\tfrac{9}{14}\lg\tfrac{9}{14} - \tfrac{5}{14}\lg\tfrac{5}{14} = 0.94.$$

Since attribute A has five 0's with 2 positive cases and 3 negative cases, four 1's with 4 positive cases and 0 negative cases, and five 2's with 3 positive cases and 2 negative cases, we can compute the $info_A$ to be

$$info_A(\text{T}) = \tfrac{5}{14}(-\tfrac{2}{5}\lg\tfrac{2}{5} - \tfrac{3}{5}\lg\tfrac{3}{5}) + \tfrac{4}{14}(-\tfrac{4}{4}\lg\tfrac{4}{4} - \tfrac{0}{4}\lg\tfrac{0}{4}) + \tfrac{5}{14}(-\tfrac{3}{5}\lg\tfrac{3}{5} - \tfrac{2}{5}\lg\tfrac{2}{5}) = 0.694.$$

The other terms associated with A are

$\text{gain(A)} = 0.94 - 0.694 = 0.246,$

$\text{split\_info(A)} = -\tfrac{5}{14}\lg\tfrac{5}{14} - \tfrac{4}{14}\lg\tfrac{4}{14} - \tfrac{5}{14}\lg\tfrac{5}{14} = 1.5774,$ and

$\text{gain\_ratio(A)} = 0.246 \;/\; 1.5774 = 0.1560.$

Computing similar values for B results in

$info_B(\text{T}) = \tfrac{6}{14}(-\tfrac{3}{6}\lg\tfrac{3}{6} - \tfrac{3}{6}\lg\tfrac{3}{6}) + \tfrac{8}{14}(-\tfrac{6}{8}\lg\tfrac{6}{8} - \tfrac{2}{8}\lg\tfrac{2}{8}) = 0.892,$

$\text{gain(B)} = 0.94 - 0.892 = 0.048,$

$\text{split\_info(B)} = -\tfrac{6}{14}\lg\tfrac{6}{14} - \tfrac{8}{14}\lg\tfrac{8}{14} = 0.9853,$ and

$\text{gain\_ratio(B)} = 0.048 \;/\; 0.9853 = 0.0489.$

- input: a training set T and an instance I to classify.
- output: a class for the instance I.

1. If T is consisted of one class, return the class.
2. If all instances have the same attribute values, return the majority class.
3. Select an attribute X with the highest score on the splitting measure, and let x be the test value on instance I.
   Assign the set of instances with X = x to T, and apply the algorithm to T.

Figure 3.12. The Algorithm for Lazy Decision Trees

Since the gain_ratio(A) is higher, attribute A is chosen as the root attribute of developed decision tree. In other words, A does a better job than B in classifying the values into the cases where classes are +, and -.

3.3.2 Lazy Decision Trees

Lazy algorithms like the nearest neighbor algorithms do not create a concise representation like a rule set or tree. Instead, they use the intact training set to find a most similar case with the input instance. Inspired by this idea, the lazy decision tree algorithm creates a tree for each test case to cope with the problems of eager decision tree algorithms (for example, C4.5, ID3, CART, and so on): fragmentation problem and difficulty in the treatment of unknown attribute values. Fragmentation problems occur at the lower part of the tree. As a tree is being built, each branch starts having less objects, so the reliability of each branch becomes worse than the upper branches. The basic idea of lazy decision tree is to build a best decision tree or path for each test case because it does not lose objects to other branches; thus avoiding the fragmentation problem. Because it does not need to test on unknown values of the test instance, the unknown value problem is easily solved. The recursive algorithm for lazy tree generation is given in Figure 3.12.

Because the entropy difference between parent and child can be negative, pure entropy of a node is not used as a spliting measure. Instead, a normalized entropy measure that assigns parent node logk, where k is the number of classes, to make parent entropy maximum. Note that if we have equal probability for each class, the entropy becomes a maximum which is equal to logk.

Because a tailored path is built for each input instance to be classified, missing attribute values are easily handled by never considering the splits on the attributes with unknown values.

All in all, better test results have been reported using this method rather than the eager algorithms [13].

3.4 Some Speculation on Rough Set Theory and Decision Trees

Several researchers have compared rough set theory with the decision tree methods [44, 73]. According to the experiment by Teghem [73], rule sets generated from each method were similar with some minor exceptions.

The good points of rough set theory are

1. its mathematically sound capability to distinguish certain from uncertain information,
2. its ability to develop rule sets having a simplified and clear interpretation, and
3. its independence from any statistical assumptions.

While rough set theory has these advantages, it is not certain how to deal with objects in boundary region, but the decision tree method has more research results for the treatment of uncertain rules.

Therefore, if we combine the entropy measure of decision trees with the good classification measure using rough set theory, we may have the ability to develop a smaller rule set size with better classification accuracy.

In the next chapter we examine how to determine a good decision attribute among several candidates in database tables.

CHAPTER 4
FINDING A GOOD DECISION ATTRIBUTE

4.1 Introduction

Because a database table is not made with the expectation for data mining, it may be unclear which is a good decision attribute. One approach would be to select a few attributes as candidates and exploit domain knowledge, which is a major ingredient of the conceptual clustering algorithms [40], to avoid the cost of constructing background knowledge. Alternatively, if we already created a training example set through some data selection process, our suggesting algorithm can be applied.

There are two factors that determine the preciseness and complexity of a rule set depending on a chosen decision attribute among possible candidate decision attributes—the level of consistency and the level of simplicity. The level of consistency refers how many consistent objects exist in terms of their decisions in a database table. The level of simplicity refers how many short rules with acceptable confidences exist in a database table.

4.2 A Method to Determine a Good Decision Attribute

To select a good decision attribute, we should check the consistency on data and the simplicity of the generating rule set for each candidate.

Checking consistency includes the task of selecting the attribute which has the largest positive region and the smallest boundary region based on rough set theory since such attribute will result in more accurate rules. But to prevent preferring many-valued attributes, which may be harmful in finding accurate rules, we give some bias.

Table 4.1. An Extreme Example Database Decision Table

A	B	C	Decision
a_1	b_1	c_1	d_1
a_1	b_1	c_1	d_2
..	..	..	..
..	..	..	..
a_1	b_1	c_1	d_n

Table 4.2. A Database Decision Table

A	B	Candidate Decision Attribute 1	Candidate Decision Attribute 2
a_1	b_1	d_1	d_1
a_1	b_2	d_1	d_2
a_2	b_1	d_1	d_3
a_2	b_1	d_2	d_4

Conflicts between rules happen when we have different decision values for the same condition. If a decision attribute has a key-like characteristic, it is highly possible that more conflicts will occur in decisions. For a set of data, a candidate decision attribute with key-like characteristics has higher possibility of conflicts than other candidates. For example, suppose we decide to select the key attribute in a database table as the decision attribute. Because a unique key value is assigned for each tuple, we have a larger chance to have conflicting decisions. See the extreme example database decision table given in Table 4.1. Note that the decision values are from key values. In rough set theory terminology, we have no positive region in this case.

We want to minimize each rule size as much as possible, but such conflicts will hinder the minimization. So, a decision attribute which has many values should be avoided to have a more consistent table of information. As another example, suppose we have the database decision table given in Table 4.2. We only have attribute B's

value for a test instance, which is b_1. We conclude d_1 with 66% confidence in the case of the first candidate decision attribute. On the other hand, we conclude d_1 with 33% confidence in the case of the second candidate decision attribute.

In addition to the consistency, to generate a simpler decision tree or rule set, we also consider the complexity of the trees. Note that generating the smallest decision tree is an NP-complete problem. So, we may be satisfied with a "good" decision tree or rule set for the target database. The considerations result in the following equation:

$$k_1 \times (\text{computed value based on the size of positive region})$$
$$+ k_2 \times (\text{computed value based on decision tree complexity})$$
$$\text{where } k_1 + k_2 = 1.$$

Depending on the user's preference, one may increment or decrement the k_1 and k_2 values. If k_1 is increased, categorization accuracy is emphasized, and if k_2 is increased, a simpler rule set is emphasized. Once approriate values are selected for k_1 and k_2, we evaluate all attributes and select an attribute having the largest value of the above equation as a decision attribute.

For example, rough set-based systems may perfer a larger k_1 value since we can find a minimal rule set for objects in the positive region, while decision tree-based systems may prefer a larger k_2 value since it will find a "good" and small rule set for the whole data.

One of the main reasons why we consider the positive region importantly is that since the target domain of KDD is usually very large databases, there is a high possibility that consistent and longer rules have a sufficient number of supporting tuples so that the rules are reliable. Note that the domain of machine learning usually has a very small number of training examples compared to the data space because they are prepared by humans. It is important to make the generated rules

as simple as possible to cover more examples so that the rule set will have a lower error rate for unseen cases.

The following are some examples to illustrate guide lines for determining the k_1 and k_2 values. First, since a high consistency means that the database table has many objects with no contradiction in decision, the size of positive region is large. If many duplicate tuples exist and/or many tuples share their attribute values with other tuples in chunk, and rather lengthy rules are tolerated by users for very high confidence, the size of positive region is important. The minimization of such rules can be done by finding the smallest size of the condition attributes' values that do not make conflicting decisions with any other objects. Here we consider all objects in the database table whether an object is in the positive region or boundary region. Note that the attribute values of the objects in the boundary region cannot be minimized. They need more attribute values for them to belong to positive region. Moreover, if the sizes of the positive region are similar among candidate decision attributes, we may prefer a decision attribute which leads a simpler rule set. So, $k_1 = 0.6 \sim 0.7$, $k_2 = 0.3 \sim 0.4$ may be used.

Second, suppose that the formation of tuples is similar to the first case. If there are big differences in the sizes of the positive regions between candidate decision attributes and we prefer a simpler rule set with acceptable confidence, we had better give less weight to the size of positive region because a larger positive region does not guarantee a simpler rule set. In this case, we may give more weight to the simpler rule set. So, $k_1 = 0.3 \sim 0.4$, $k_2 = 0.6 \sim 0.7$ may be used. In addition, if a more accurate rule set is desired, we should give a smaller error rate value as the threshold for the trial decision tree generation.

Third, if there are a variety of tuples and their values are not similar, we may prefer simpler rule sets since lengthy rules will be less reliable. So, $k_1 = 0 \sim 0.1$, $k_2 = 0.9 \sim 1$ may be used.

Fourth, if we want to apply rough set theory to find a minimal rule set, the simplicity of the rule set is also important. So, $k_1 = 0.5$, $k_2 = 0.5$ may be used.

One way to determine the "chunks" is to use the modified association rule finding algorithm presented in Chapter 5 and 7. If the size of target database is very large, sampling can do the job well enough.

4.3 Computing The Score Based On Decision Tree Complexity

If a database is large, computing the decision tree complexity-based value will take significant amount of computing time. To limit this time we may specify a threshold value as the error rate of the trial decision tree. Then we can examine the error rate of the trial decision tree as the tree is being generated in a BFS (Breadth First Search) manner to avoid the time consuming pruning task. We may also terminate the generation of each branch if a node satisfies a certain threshold of error rate to distribute the pseudo pruning effect on the entire tree. We use the term pseudo pruning because we do not prune the tree explicitly for trial decision trees. The error rate is computed by applying Quinlan's PEP (pessimistic error pruning) [20] method to take advantage of its top down pruning and simple computation while beholding its good performance. The error rate of the tree being generated is

$$\frac{\sum_{t \in L}(e(t) + 0.5)}{\sum_{t \in L} n(t)}$$

where L is the set of leaf nodes, e(t) is the number of objects not belonging to the majority in leaf node t, and n(t) is the number of objects in leaf node t.

The score based on the complexity of tree is adjusted to assign a larger score to the simpler tree. The simplest tree receives a score of 100, and other trees receive scores based on their relative complexity using the following equation:

100 / {(a tree size)/(the simplest tree size)}.

4.4 Computing The Score Based on The Size of Positive Region

Suppose we have a set of objects and the attribute set in a database table which describes the objects, and we can add attributes as needed. The table has one decision attribute and other attributes are condition attributes. In this case, as the number of attributes in the database table becomes larger, the positive region becomes at least the same or larger than the database table with a smaller number of condition attributes. The correctness of this property is given by the following theorem.

Theorem. If condition attributes are added to an existing decision table, the positive region of the table becomes equal to or larger than the previous table.

Proof. Because adding attributes to existing condition attributes plays the role of at least subdividing the regions that are made with the previous condition attributes, the previous positive and boundary region are subdivided. Because the subdivision of the existing positive region does not make any object belong to the boundary region, the positive region is at least maintained as before and, moreover, the positive region will be increased if some objects in the boundary region are classified without contradiction by the extra condition attributes.

On the otherhand, if the positive region is diminished when condition attributes are added, some part of the positive region with the previous condition attributes must belong to the boundary region. Thus, this contradicts with the definition of positive region. (Q.E.D.)

So, when we apply the positive region checking method to a database table, using the maximum number of condition attributes available is the best method to determine the positive region of the given database table. How many of them is used in each individual rule is dependent on the rule generation algorithms. For example, if we use rough set theory, we can find a minimal rule set for objects that belong to the positive region as illustrated in Chapter 3. To compute the value based on the size of the positive region, repeat the following steps for each user-selected potential decision attribute.

Steps:

1. Select an attribute.
2. Sort the table using all attributes of the table other than this decision attribute.
3. Find the size of the positive region: count the number of rows having the same value on the condition attributes but having different values on the decision attribute. Note that this number is the number of rows belonging to the boundary region. By subtracting the value from the total number of rows, we obtain the number of rows belonging to the positive region.

If an attribute has key-like characteristics, it will have a very small boundary region, so it will be harmful to rule set minimization. To decrease the chance that these attributes will become decision attributes, we need some adjusting factor to score the attributes as the denominator of the following equation:

$$(\frac{|X|}{|T|})/(\frac{|D|}{|B|})$$

where

- X = the number of rows belonging to the positive region,
- T = the total number of rows of the table,
- D = the number of values (classes) possessed by the decision attribute currently being considered, and
- B = the number of values of the decision attribute which has the largest number of values among all potential decision attributes.

The score based on the positive region is adjusted to give the ideal decision attribute 100. The numerator of the equation has the value range of $0 \sim 1$; the largest value is 1, the middle is 0.5, and the smallest is 0. The denominator of the equation has the largest value of $|B|/|B| = 1$, the middle value of $\{\ (|B| + |C|)/2\ \} / |B|$, and the smallest value of $|C|/|B|$, where B is defined as above and C is the number of values of the decision attribute which has the smallest number of values among all potential decision attributes. So, the smallest and largest value of the equation are

$$x_0 = 0, \text{ and}$$
$$x_2 = 1 \ / \ (\tfrac{|C|}{|B|}).$$

Considering both of the numerator and denominator for middle point, we get

$$x_1 = 0.5 \ / \ (\tfrac{(|B|+|C|)/2}{|B|}).$$

The score is assigned by the linear interpolation of three points x_0, x_1, and x_2. The resulting scores for x_2, x_1, and x_0 are 100, 50, and 0 respectively. So, the score y for some x is

$y = 50\{x/(x_2 - x_1) + 1 - x_1/(x_2 - x_1)\}$ when $x_1 \leq x \leq x_2$,
$y = 50(x/x_1)$ when $0 \leq x \leq x_1$.

Obj.#	a	b	d	e	f
1	0	0	1	1	1
2	0	1	2	2	2
3	0	2	2	2	3
4	1	0	3	1	4
5	1	1	1	1	5
6	1	2	2	1	6
7	1	1	2	2	5
8	1	2	1	1	7

Figure 4.1. A Relational Table

Example. Consider the relational table in Figure 4.1 where the condition attributes are a and b, and the candidate decision attributes are d, e, and f.

Although decision = 1 is made by objects 1, 5, and 8 when the decision attribute is d, objects 5 and 8 cannot belong to positive region since object 5 conflicts with object 7 which has decision = 2 and object 8 conflicts with object 6 which has decision = 2.

So, $\underline{C}D_1 = \{1\}$.

Similarly, $\underline{C}D_2 = \{2, 3\}$,

$\underline{C}D_3 = \{4\}$,

and, $\underline{C}E_1 = \{1, 4, 6, 8\}$,

$\underline{C}E_2 = \{2, 3\}$,

and, $\underline{C}F_1 = \{1\}$,

$\underline{C}F_2 = \{2\}$,

$\underline{C}F_3 = \{3\}$,

$\underline{C}F_4 = \{4\}$,

$\underline{C}F_5 = \{5, 7\}$,

$\underline{C}F_6 = \emptyset$, and

$\underline{C}F_7 = \emptyset$.

Obj.#	a	b	d	e	f
1	0	0	1	1	1
2	0	1	2	2	2
3	0	2	2	2	3
4	1	0	3	1	4
5	1	1	1	1	5
7	1	1	2	2	5
6	1	2	2	1	6
8	1	2	1	1	7

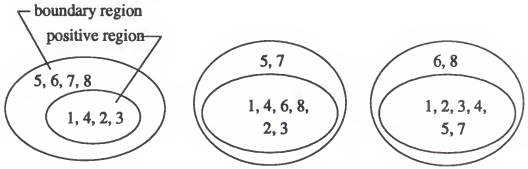


Figure 4.2. An Example Table Sorted by Condition Attribute Values and the Corresponding Diagrams

See the example table sorted by condition attribute values and the corresponding diagrams in Figure 4.2. If we represent the table in rule form, we have rules as follows:

When the decision attribute is D,

for object 1: $a_0b_0 \rightarrow d_1$,

for object 2: $a_0b_1 \rightarrow d_2$,

for object 3: $a_0b_2 \rightarrow d_2$,

for object 4: $a_1b_0 \rightarrow d_3$,

for objects 5,7: $a_1b_1 \rightarrow \{d_1, d_2\}$, and

for objects 6,8: $a_1b_2 \rightarrow \{d_1, d_2\}$.

When the decision attribute is E,

for object 1: $a_0b_0 \rightarrow e_1$,

for object 2: $a_0b_1 \rightarrow e_2$,

for object 3: $a_0b_2 \rightarrow e_2$,

for object 4: $a_1b_0 \rightarrow e_1$,

for object 6: $a_1b_2 \rightarrow e_1$,

for object 8: $a_1b_2 \rightarrow e_1$, and

for objects 5,7: $a_1b_1 \rightarrow \{e_1, e_2\}$.

And when the decision attribute is F,

for object 1: $a_0b_0 \rightarrow f_1$,

for object 2: $a_0b_1 \rightarrow f_2$,

for object 3: $a_0b_2 \rightarrow f_3$,

for object 4: $a_1b_0 \rightarrow f_4$,

for object 5: $a_1b_1 \rightarrow f_5$,

for object 7: $a_1b_1 \rightarrow f_5$, and

for objects 6,8: $a_1b_2 \rightarrow \{f_6, f_7\}$.

So, when the decision attribute is D, $\frac{4}{8}/\frac{3}{7} = 1.1666$,
when decision attribute is E, $\frac{6}{8}/\frac{2}{7} = 2.6251$, and
when decision attribute is F, $\frac{6}{8}/\frac{7}{7} = 0.75$.

The normalized scores for each candidate decision attribute are $50\{1.1666$ / (3.5002 - 0.7777) + 1 - 0.7777 / (3.5002 - 0.7777)$\} = 57.14$ for attribute D, and similarly 83.93 for attribute E, and $50(0.75$ / $0.7777) = 48.22$ for attribute F, where $x_2 = 3.5002$, $x_1 = 0.7777$, and $x_0 = 0$. Since E has the largest value, if we select it as the decision attribute, we will have less conflict. Note that attribute F has key-like characteristics, that is, it has smallest computed value, so it cannot be selected as the decision attribute.

4.5 Preprocessing the Table

Because our method of finding frequent patterns and converting them into rules only processes symbolic values, we need to convert numeric values into symbols. For example, if our database contains the height of individuals in centimeters, we could convert these height values into the nominal values of very_tall ($\geq$ 200 cm), tall (180 – 200), normal (180 – 170), slightly_small (160 – 170), small (150 – 160), and very_small (150 $\leq$).

Some researchers have tried to find good interval structures based on statistics [59, 68, 78], and while finding a good intervals for numeric values is a research area, the intervals are prone to be subjective. So, making use of the user's background knowledge and brevity for implementation, we will provide appropriate intervals for numeric values. A simple way to make appropriate intervals is to use Sturges' criteria:

$$\mathrm{n} = 1 + log_2 \mathrm{N}$$

where n is the number of intervals and N is the number of data values.

4.6 Time Complexity of the Algorithm

Selecting a decision attribute from the table consists of two substeps. The first one is sorting, which is $O(|r|lg|r|)$ where $|r|$ is the number of rows. The second one is making trial decision trees. The cost at each node of the tree is $O(|r||A|)$ where $|A|$ is the number of attributes. It has been known that the induction does not exponentially grow in time and space [62]. Moreover, the trial decision tree generation continues only until a given pessimistic error rate.

4.7 Experimentation

We used a real world data set from the UCI machine learning data repository [54]. A database called "adult" containing 32,561 objects for training and 16281 objects

for testing was chosen giving a total number of objects of 48842. Originally the data set was extracted from the 1994 census database. It has six continuous attributes and eight nominal attributes. Attributes age, fnlwgt, education-num, capital-gain, capital-loss, and hours-per-week are continuous attributes, while workclass, education, marital-status, occupation, relationship, race, sex, and native-country are nominal attributes. Conversion of the six continuous attributes to nominal values was performed as follows:

1. Attribute age is divided into twelve nominal values: below-10 (represented by 5), teens (10), twenties (20), thirties (30), ..., and more-than-100 (100).
2. Attribute hours-per-week is divided into the intervals: below-10, 10-20, 20-30, and more-than-40.
3. Attribute education-num is divided into the intervals: less-than-two-years (1), less-than-four-years (3), ..., less-than-fourteen-years (13), less-than-fifteen-years (14), less-than-seventeen-years (16), ..., less-than-twenty-one-years (20), and more-than-twenty-one-years (21).

The other three continuous attributes (fnlwgt, capital-gain, and capital-loss) are divided into 16 intervals based on Sturges' criteria. There are two class values, the yearly income of less than \$50,000 (class 1), and more than \$50,000 (class 2).

For our experiment we wanted to check if attribute *sex* is a better decision attribute than the original decision attribute, yearly_income. The number of objects belonging to positive region under the original decision attribute is 38,156 while the size of positive region when sex is decision attribute is 41,474. So, according to the size of positive region, attribute sex is better than the other attribute.

For original decision attribute,

$$(\frac{38156}{48842})/(\frac{2}{2}) = 0.7812.$$

For decision attribute sex,

$$(\frac{41474}{48842})/(\frac{2}{2}) = 0.8491.$$

Trial decision trees were made up to the pessimistic error rate of 16% to save computing time by using the training data of 32561 objects. For the original decision attribute the total number of nodes is 316, while the total number of nodes for sex is 4712. So, the original decision attribute has a smaller decision tree even though it has smaller positive region.

The normalization is done as follows:

the scores based on positive region:
for the original: $50 \times \{0.7812/0.5 + 1 - 0.5/0.5\} = 78.12$.
for sex: $50 \times \{0.8491/0.5 + 1 - 0.5/0.5\} = 84.91$.

the score based on decision tree complexity:
for the original: $100/(316/316) = 100$.
for sex: $100/(4712/316) = 6.7$.

Because the adult database has many "chunks" of frequent patterns (see Chapter 5 and 7), let $k_1 = 0.7$ and $k_2 = 0.3$, the score of original decision attribute is

$$0.7 \times 78.12 + 0.3 \times 100 = 84.68.$$

The score of decision attribute sex is

$$0.7 \times 84.91 + 0.3 \times 6.7 = 61.45.$$

Therefore, as we expected, the original one is a better decision attribute. The result suggests the available data may be biased so that the level of consistency or the size of positive region may not lead to a simpler rule set. Note that because we can

Table 4.3. The Number of Nodes generated with Pessimistic Error Rate of 18% for Random Data

	Number of Nodes
Original data	73
20% conflict	1051
10% conflict	138

find a minimal size rule set in positive region by applying rough set theory, we may have a simpler rule set if we consider objects belonging to the positive region only. Therefore, the result of experiment implies that because pruning of decision trees ignores some minor change of error rate for simpler trees, the dependency between each individual values of condition attributes on decision attribute values, and how such dependencies exist in "chunks" determine the overall complexity of the decision tree generated.

To make this point clear, we created random data by using the *dgp2* data generation program in the UCI machine learning repository since the program generates random data in normal distribution with no conflicting classifications. Condition attributes are dependent only on the decision attribute which has positive and negative as its values. To make conflicting decision values after generating 50,000 objects, duplicate objects were made. From it randomly 20% of the objects were made to have conflicting decision values, and similarly 10% of the objects were made to have conflicting decisions. The size of positive region is reduced from 100,000 to 79,644 and 89,876 respectively. The size of trial decision trees when the pessimistic error rate is 18% is given in Table 4.3.

If we compute scores based on positive regions like before, we get

Table 4.4. The Number of Nodes generated using C4.5 with Estimated Error Rate of 14.4% for Random Data

	Number of Nodes
Original data	713
20% conflict	11451
10% conflict	4000

Table 4.5. The Number of Nodes generated using C4.5 with Estimated Error Rate of 15.8% for Random Data

	Number of Nodes
Original data	377
20% conflict	7517
10% conflict	2219

for original data: $100000/100000 = 1 \Rightarrow 100$,

for 20% conflict: $79644/100000 = 0.79644 \Rightarrow 79.644$, and

for 10% conflict: $89876/100000 = 0.89876 \Rightarrow 89.876$.

The scores based on the size of trial decision trees are computed like

for original data: 100,

for 20% conflict: $100/(1059/73) = 6.9$, and

for 10% conflict: $100/(138/73) = 52.9$.

Since one has a 100% positive region, we may want to apply rough set theory to find a minimal rule set, so let $k_1 = k_2 = 0.5$, the scores are 100, 43.27, and 71.4 for original data, 20% conflict, and 10% conflict respectively.

The result reveals the fact that the bigger the size of positive region is, the smaller the decision tree is on the condition that data values are uniformaly dependent on decision values. Actual decision trees for this random data were generated using C4.5 [63] with the estimated error rate of 14.4% and 15.8% as shown in Table 4.4 and Table 4.5 respectively.

Table 4.6. The Number of Nodes generated using C4.5 with Estimated Error Rate of 16.4% for Adult Data

Decision Attribute	Number of Nodes
Original	250
Sex	1364

The summary of the generated decision tree for adult data using C4.5 is shown in Table 4.6 when the estimated error rate is 16.4%. From Table 4.6 we can determine that since pruning is performed to cut branches that do not affect overall error rate much, the original decision attribute in the adult database has a strong dependence on some smaller set of values that have high influence on the decision. On the other hand, the result of the random data experiment reveals the fact that when data are unbiased with respect to attributes, the size of decision tree is proportional to the overall positive region.

Because the domain of KDD is generally large databases depending on the target applications where the rules are to be used, one may prefer a rule set in which each individual rule is rather lengthy, very confident, and has a sufficient number of supporting objects, rather than a simpler rule set with smaller overall error rate, and visa versa. Note that the complexity of the rule set is also very dependent on the pruning rate in decision trees [6]. For example, the number of nodes when pruning confidence 95% for each candidate decision attribute in adult data using C4.5 is shown in Table 4.7. Note that the default pruning confidence in C4.5 is 25%. The smaller the value is, the severe the pruning is. So, C4.5 system becomes less confident about the resulting tree as the pruning becomes severe. One can give $0.001\% \sim 100\%$ as a parameter. So, an appropriate error rate as a threshold should be given to compare the simplicity of each decision tree.

Decision Attribute	Number of Nodes	Estimated Error Rate
Original	9169	12.1%
Sex	12835	11.8%

Table 4.7. The Number of Nodes generated using C4.5 for Adult Data

4.8 Conclusions

Due to limited domain knowledge users may not be sure which one is the best decision attribute among several possibilities in a database table. To provide users an objective assessment measure in selecting a good decision attribute depending on the property of target data and users' need, our method uses two factors—the level of consistency and the simplicity of rule sets. The level of consistency is measured by the size of the positive region in terms of rough set theory and the simplicity of the rule sets is measured by trial/actual decision trees. So, by considering both factors in determining a good decision attribute among possible candidates, one may get a better rule set for his/her own purpose. Moreover, even if one prefers an attribute as a decision attribute, our method can provide him/her the information on how good it is, compared to other possible decision attribute. One may generate a rule set by using his/her favorite method after a decision attribute is chosen.

In the next chapter we present a method of stepwise refinement of rule set based on frequency that can allow users to find an optimal rule set for their interests.

CHAPTER 5
STEPWISE RULE SET DISCOVERY

5.1 Introduction

A real world database contains numerous manifest facts. So, the task of KDD systems is comparable to finding single grains of gold in the sand on a beach. While many KDD systems are based on some decision tree generation algorithms, decision trees [5, 55, 63] have their own weak points. Since most decision trees are created to minimize the prediction error rate, the generation algorithms prefer shorter trees where the measurement used to select the root node of each subtree excludes domain knowledge. As a result, these trees tend to discover knowledge that may not be useful [46, 61, 72]. Compounding this program is the size of the database. If the target database is very large, the generated decision tree can be very large resulting in a comprehensibility problem.

In a genetic algorithm-based machine learning system like GABIL [16], the correctness of a population (that is, rule set being trained) to the training examples plays very important role in the evolution process. Therefore, we can think of the system as a pattern finding system which also relies on randomized search. The weak points of most genetic algorithm-based learning systems are their computational intensiveness if the size of training example is large and they only allow positive/negative classes. If the purpose of KDD is description finding, our method proves to be a good alternative. If we take advantage of the fact that more general concepts occur more frequently, we may eliminate the manifest facts from the discovered knowledge and focus on interesting facts in a stepwise manner [71].

5.2 An Overview of Stepwise Refinement

In the early stage of stepwise refinement, we may specify a very large minimum support number which may result in counting only those items occurring in 10% or more in the whole database. We generate rules based on the found patterns that occur more than the given minimum support number. After inspecting the generated rules, the users can determine which are the interesting/uninteresting rules. If the users wish to see a single hierarchical structure of potential rules to grasp a general view of the domain, they may generate a trial decision tree in breadth first search manner with some specified depth limit [70]. Note that due to the combinatorial nature of our method of rule set generation, making a hierarchical structure of found rules is difficult unless some bias like users' interests is applied.

Once users have selected interesting rules, the next step of the refinement focuses on finding rules whose condition parts are a superset of the chosen interesting rules. If, however, the users have selected uninteresting rules, we can nullify items from each row of the original database. This causes these items and their related rules not to appear in the next step of the stepwise refinement resulting in a smaller size rule set in the next step.

5.3 The Modified Association Rule Finding Algorithm

We devised an optimal rule set finding algorithm for frequently occurring cases based on the following definitions.

Definitions.

1. Database decision table: a table having attributes with condition and decision parts to represent information.

2. Item: each distinct attribute-value pair of the table. All values are nominal.

3. Itemset: any combination of items in each row of the database decision table. Note that due to the property of the table an attribute can appear only once in the itemset, so items with different attributes constitute a different itemset.

4. Frequency(X): how many times an itemset X occurs in a table.

5. Minimum support number: a threshold to bisect itemsets into frequent and infrequent itemsets based on their frequency.

6. Frequent itemset: an itemset occurring more than the given minimum support number.

Note that the above definitions are a specialization of the terms defined for the association rule finding algorithms which originally were developed to find purchasing patterns from transactional databases. Our algorithm is an improved version of these algorithms that applies to the database decision tables.

Another difference with the original algorithms is in the construction of itemsets. Each itemset consists of a set of attribute-value pairs with an attribute appearing only once in the itemset.

The efficiency of the original algorithms in finding frequent patterns from large transactional databases is accomplished by the fact that each combination of N-1 items in size N itemsets of iteration N exists in some size N-1 itemsets of iteration N-1. It has been reported that the algorithms have reasonable response time [2]. It takes minutes of processing time on a workstation to execute the algorithm for a transactional database consisting of several million records.

Our algorithm is a modified version of the Apriori algorithm [2], so depending on the available hardware resources, one may apply a more efficient association rule finding algorithm like AprioriTid [2], a hash-table based association rule finding algorithm [57], or even a parallel version of the algorithm [65]. Even though our algorithm

```
F_1 = { frequent 1-itemsets}
For k=2 till the given itemset length do
      C_k = generate_candidates(F_{k-1} )
      For all rows r in the database decision table do
            For all combinations c of size k using r
                  Match c to candidate c' in C_k
                  If matched then
                        Increment counter of c'
                  endif
            endfor
      endfor
F_k = {c' in C_k | the counter of c' ≥ min_support_no}
endfor
```

Figure 5.1. The Modified Association Rule Finding Algorithm (part 1)

is a modified version of the Apriori algorithm, we do not need to generate itemsets exhaustively and the user may specify the maximum desired length of rules for each step. See the modified association rule finding algorithm (part 1) in Figure 5.1.

The function generate_candidates(F_{k-1}) takes all frequent (k-1) itemsets F_{k-1} as input and returns all possible frequent k itemsets, each of which has distinct attributes for each item. See the modified association rule finding algorithm (part 2) in Figure 5.2.

5.4 Rule Set Generation

There are four principles to remember when we generate rules.

First, give priority to refined rules. If a rule is a member of the subset of the refined interesting rules, that rule may not be presented to users. We may select rules that have more than a given confidence threshold on the delegate rules. A delegate rule is the rule of the highest confidence among the rules having the same condition part but different decision parts.

generate_candidates(F_{k-1}):
For all itemset in F_{k-1} do
    Select I_{item1}, I_{item2}, ... , $I_{item(k-1)}$, $J_{item(k-1)}$,
    From F^I_{k-1}, F^J_{k-1} // F^I_{k-1} means the ith itemset in F_{k-1}
    Where I_{item1}=J_{item1}, ... , $I_{item(k-1)} \neq J_{item(k-1)}$, and
        attribute($I_{item(k-1)}$) $\neq$ attribute($J_{item(k-1)}$)
    Insert the itemset into C_k'
endfor
For all itemset c' in C_k' do // prune step
    if all combinations of size (k-1) using c' exists in F_{k-1} then
        Insert the itemset c' into C_k
    endif
endfor

Figure 5.2. The Modified Association Rule Finding Algorithm (part 2)

Second, if two rules A and B are the refinements of another rule C (so they have the same decision) and A's condition is a superset of B's, but A's confidence is less than B's, then A should be pruned. For example, suppose we have found a rule (in parenthesis is the rule name), (C) a1 $\rightarrow$ d1 and we have two refined rules after a refinement like

(B) a1 b1 $\rightarrow$ d1 (70%), and

(A) a1 b1 c1 $\rightarrow$ d1 (50%).

Then the second rule should be pruned. If their origin is different, only the fact is indicated to the longer rule, and which rule will be preferred is user's choice. For example, suppose we have found a rule set

a1 c1 $\rightarrow$ d1 (1)

b1 $\rightarrow$ d1 (2)

and we have the following two refined rules:

$$a1\ \ b1\ \ c1 \rightarrow d1 \quad (3)$$
$$b1\ \ c1 \rightarrow d1 \quad (4).$$

Rule (3) is a refinement of rule (1) and (2), and rule (4) is a refinement of rule (2). If the confidence of rule (3) is less than rule (4), rule (3) may be pruned. But, since rule (3) is the refinement of rule (1), we will display the rule and allow the user to choose his favorite. The user might want to refine rule (3) only because he might be interested in rule (3), but not rule (4). If the rule is not selected, the rule should be removed for the next step. Because the confidence of rule (3) is $|a1\ b1\ c1\ d1|$ / $|a1\ b1\ c1|$ and the confidence of rule (4) is $|b1\ c1\ d1|$ / $|b1\ c1|$, and $|a1\ b1\ c1| \leq |b1\ c1|$ and $|a1\ b1\ c1\ d1| \leq |b1\ c1\ d1|$, so rule (3)'s confidence can be larger than rule (4)'s, and visa versa.

Third, generated rules should be sorted based on (1) the condition attribute values and (2) the decision attribute values. Rules with the same condition part and different decision parts are presented together so the user can easily determine the properties of the found rule set for further stepwise refinement.

Fourth, when the minimum support number is small, which usually occurs in the later step of the refinement, rules with a small confidence may be dropped as uninteresting.

5.5 The Stepwise Refinement Algorithm and Properties of Refinement

The refinement progresses as detailed in Figure 5.3. Note that if interesting rules are selected in the previous steps, rules with a superset of the condition part of these rules only may be presented to the users in the current step.

When we supply a minimum support number, some objects are completely excluded. So, depending on the user's selection in each step of the refinement, some objects are never considered. For example, suppose we have the database decision

Repeat

1. Run the modified association rule finding algorithm with an appropriate minimum support number and rule length.
2. Inspect the generated rules and determine interesting/uninteresting rules.
3. If interesting rules are selected then
   (a) select all objects that have the same condition part with the interesting rules and
   (b) make a subset of the original database decision table.
4. If uninteresting rules are selected then
   nullify attribute values that are contained in the selected rules.

Until no need for further refinement.

Figure 5.3. The Stepwise Refinement Algorithm

table of Figure 5.4 where we have attributes A and B as condition attributes, and Decn as the decision attribute.

If the minimum support number is 4, the rule

$$a1 \rightarrow d1 \ (71\%)$$

can be found.

If we select this rule as interesting, objects 1 through 7 will be selected for the next stepwise refinement. If the next minimum support number is 2, we may find other rules like

$$a1\ b1 \rightarrow d1 \ (100\%),$$
$$a1\ b2 \rightarrow d1 \ (100\%), \text{ and}$$
$$a1\ b3 \rightarrow d2 \ (67\%).$$

Note that in this example, objects 8-10 are completely excluded, and rules like

$$b1 \rightarrow d1 \ (100\%),$$

Object #	A	B	Decn.
1	a1	b1	d1
2	a1	b1	d1
3	a1	b2	d1
4	a1	b2	d1
5	a1	b3	d2
6	a1	b3	d2
7	a1	b3	d2
8	a2	b4	d3
9	a2	b4	d3
10	a2	b1	d3

Figure 5.4. A Database Decision Table

$$b2 \rightarrow d1 \text{ (100\%), and}$$

$$b3 \rightarrow d2 \text{ (67\%)}$$

may not be presented since they are subsumed by the above rules.

On the other hand, if the initial minimum support number is 3, we may find rules like

$$a1 \rightarrow d1 \text{ (71\%)},$$

$$a2 \rightarrow d3 \text{ (100\%)},$$

$$b1 \rightarrow d1 \text{ (67\%), and}$$

$$b3 \rightarrow d2 \text{ (67\%)}.$$

If we decide rule $a2 \rightarrow d3$ is uninteresting and the next minimum support number is 2, we may find additional rules like

$$b2 \rightarrow d1 \text{ (100\%)},$$

$$b4 \rightarrow d3 \text{ (100\%), and}$$

$$a1\ b1 \rightarrow d1 \text{ (100\%)}.$$

Note that the second rule was not found before because its dominant value, which is a2, belonged to rare cases and rules having attribute B only in the rare cases

Object #	A	B	C	Decn.
1	a1	b2	c2	d1
2	a1	b2	c3	d2
3	a1	b2	c4	d1
4	a1	b3	c1	d1
5	a2	b1	c1	d1
6	a2	b1	c5	d2
7	a3	b1	c6	d1
8	a3	b1	c7	d1
9	a2	b2	c1	d2

Figure 5.5. A Database Decision Table

were not generated. Therefore, domain knowledge is important in determining the minimum support number and in choosing interesting rules or uninteresting rules for the next step of refinement.

Another property of stepwise refinement is related to the selection of a minimum support number for each step. As a smaller minimum support number is supplied, rules that are not the refinement of rules of the previous steps can be found. The confidence of these rules may not be calculated correctly if we consider only that part of the database resulting from focusing on interesting rules. To illustrate this, consider we have the database decision table of Figure 5.5.

If the initial minimum support number is 4, we can find rules

$$a1 \rightarrow d1 \ (75\%),$$
$$b1 \rightarrow d1 \ (75\%),$$
$$b2 \rightarrow d1 \ (50\%), \text{ and}$$
$$b2 \rightarrow d2 \ (50\%).$$

If we select the first two rules as interesting and supply the next minimum support number of 2, we will find more rules like

$$a2 \rightarrow d2 \ (67\%),$$

c1 → d1 (67%),

a1 b2 → d1 (67%),

a2 b1 → d1 (50%), and

a3 b1 → d1 (100%).

Note that for the first two rules, if we think only of the selected objects, their confidences are 50% and 100% respectively (note that object 9 has been dropped). Therefore, for rules that are not a refinement of rules from the previous step, their confidence should be based on the whole database. But calculating the confidence of the rules and selecting the corresponding objects may cause significant overhead, and the rules are a biased by-product of the objects that produced interesting rules, that is, rules that can be made only from chosen objects where users did not intend to find them. So, based on the users' preference, rules that are the refinement of interesting rules only may be considered.

5.6 The Benefit of Selecting Uninteresting Rules

Suppose we have rules like

sex=female → can_have_a_baby,

sex=female & age=young → can_have_a_baby, and

sex=female & health_cond=good → can_have_a_baby.

As you see, these rules contain manifest facts. If we nullify the condition of the first rule, the other two rules will disappear. The confidences of rules like

age=young → can_have_a_baby, and

health_cond=good → can_have_a_baby

which will still remain after the nullification may not be large enough to be considered good candidate rules. To further illustrate this point, suppose we have the database

Object #	A	B	C	Decn.
1	a1	b1	c1	d1
2	a1	b1	c1	d1
3	a1	b1	c2	d1
4	a1	b1	c2	d2
5	a1	b1	c3	d1
6	a1	b2	c3	d2
7	a1	b2	c4	d2
8	a1	b2	c4	d2
9	a2	b3	c1	d3
10	a2	b3	c5	d3
11	a2	b3	c2	d3
12	a2	b1	c2	d4
13	a2	b4	c3	d4
14	a3	b3	c5	d4

Figure 5.6. A Database Decision Table

decision table in Figure 5.6. If the initial minimum support number is 5, we can find rules

$$a1 \rightarrow d1 \text{ (50\%)},$$

$$a2 \rightarrow d3 \text{ (60\%)},$$

$$b1 \rightarrow d1 \text{ (67\%), and}$$

$$a1\ b1 \rightarrow d1 \text{ (80\%)}.$$

Selecting the first rule as uninteresting, we nullify the value a1 of attribute A. If the next minimum support number is 4, we can find rules like

$$a2 \rightarrow d3 \text{ (60\%)},$$

$$b1 \rightarrow d1 \text{ (67\%), and}$$

$$b3 \rightarrow d3 \text{ (75\%)}.$$

Note that rules like

$$a1 \rightarrow d1 \text{ (50\%), and}$$

$$a1\ b1 \rightarrow d1 \text{ (80\%)}$$

are not found in the second refinement step since rule a1 $\rightarrow$ d1 (50%) has been chosen as uninteresting in the initial step. Thus, the more rules that are chosen as uninteresting, the less rules with satisfactory confidences that will be found at later refinements.

5.7 Time Complexity

The time complexity of the original association rule finding algorithm is O(k log (m/k)) where m is the number of items in the database and k is the size of the frequent itemset [2]. This is considered a reasonable response time. Our algorithm executes only a few iteration of the original algorithm in each refinement, since we do not need to generate rules exhaustively. The algorithm is also a little more efficient in each refinement, since we do not need to generate itemsets with duplicate attributes.

5.8 Experimentation

To evaluate this algorithm we used a real world data set from the UCI machine learning data repository [54]. A database called adult which contains 32,561 objects for training was chosen. Details of the database and how data conversion has been done is in Chapter 4. There are two class values, the yearly income of less than \$50,000 (class 1), and more than \$50,000 (class 2).

For the first step, we specified a minimum support number of 6000, resulting in 21 size two rules as in Figure 5.7. Here, frq means frequency, and cf means confidence.

When this set was examined, the second rule was chosen as interesting. We then proceeded to the second step with the minimum support number = 3000.

Among the 18 size three rules found, the following rule was chosen as interesting:

age = 30 & sex = male $\rightarrow$ class 1 (frq=4119, cf=68.23%).

For third step, we supplied the minimum support number of 1500, resulting in 42 size four rules. Among them, the following rule was chosen as interesting:

age = 20	→ class 1 (frq=7544, cf=93.67%),
age = 30	→ class 1 (frq=6304, cf=73.19%),
workplace = private	→ class 1 (frq=17733, cf=78.13%),
fnlwgt = 138039	→ class 1 (frq=9133, cf=74.10%),
fnlwgt = 230065	→ class 1 (frq=7452, cf=76.79%),
education = HS-grad	→ class 1 (frq=8826, cf=84.05%),
education-num = 9	→ class 1 (frq=9226, cf=84.38%),
education-num = 11	→ class 1 (frq=6924, cf=79.83%),
marital-status = married-civ-spou	→ class 1 (frq=6692, cf=44.68%),
marital-status = married-civ-spou	→ class 2 (frq=8284, cf=55.32%),
marital-status = Never-married	→ class 1 (frq=10191, cf=95.39%),
relationship = Husband	→ class 1 (frq=7275, cf=55.14%),
relationship = Not-in-family	→ class 1 (frq=7448, cf=89.68%),
race = white	→ class 1 (frq=20699, cf=74.41%),
race = white	→ class 2 (frq=7117, cf=25.59%),
sex = Female	→ class 1 (frq=9592, cf=89.05%),
sex = Male	→ class 1 (frq=15127, cf=69.42%),
sex = Male	→ class 2 (frq=6662, cf=30.57%),
capital-gain = 3124	→ class 1 (frq=24651, cf=79.26%),
capital-gain = 3124	→ class 2 (frq=6448, cf=20.73%),
capital-loss = 136	→ class 1 (frq=23978, cf=77.23%),
capital-loss = 136	→ class 2 (frq=7068, cf=22.77%),
hours-per-week = 45	→ class 1 (frq=17694, cf=71.35%),
hours-per-week = 45	→ class 2 (frq=7103, cf=28.64%),
native-country = United-States	→ class 1 (frq=21998, cf=75.42%),
native-country = United-States	→ class 2 (frq=7171, cf=24.58%).

Figure 5.7. Rule Set with Large Support Number

age = 30 & sex = male & workplace = private → class 1 (frq=3059, cf=68.9%).

For our last step, we supplied the minimum support number of 700, resulting in 135 size five rules. Among them the following rule was chosen as interesting:

age = 30 & sex = male & workplace = private & education_num = 11
→ class 1 (frq=759, cf=71.27%).

We may infer from this last rule that there might be a variety of reasons for an income of class 1 for a male of an age of thirties who worked in a private place. The most frequent reason for this income is the individual's relatively poor educational background.

5.9 Conclusions

The rule sets generated by decision tree methods are just a small part among the vast possibilities, and it is difficult to fully incorporate the users' interests in the generating decision trees. As a consequence, the resulting trees may not be as useful as expected. The genetic algorithm-based learning systems have the difficulty of devising a "good" fitness function that can reflect users' interests well since, in general, users' interests or domain knowledge is not well defined. Moreover, they require intensive computation for training. Our stepwise refinement technique considers all possibilities in generating rules with respect to a given minimum support number, so it can generate an exhaustive rule set in the direction of the users' interests. But, the stepwise refining of rule sets, which is based on the frequency of data, can allow users to find the best rule set of the users' interests by helping the users to use their ill-defined domain knowledge through stepwise refinement. Thus avoids the possibility of generating numerous rules and the problems related to the task of specifying the ill-defined nature of domain knowledge or users' interests. Another good point of our method is that the number of supporting objects of the generated descriptive rules

is large enough to be considered reliable in a statistical sense. Because of the large size of the target databases, the rules may have good prediction accuracy also.

The weak points of our method are its high computational complexity and the lack of understandability if the minimum support number in the early steps is small. Users also should be careful in selecting interesting/uninteresting rules to narrow the search space.

In this chapter we illustrated how we can find optimal interesting rules by a stepwise refinement technique. In the next chapter we present a hybrid decision tree having the ability to generate very accurate predictions and saving on tests of some attribute values.

CHAPTER 6
HDT: A HYBRID DECISION TREE

6.1 Introduction

Although decision trees are one of the most successful data mining/machine learning approaches, they may not always be the best predictor because they are built to cover all of the data in a single tree. There may be data instances that are represented awkwardly in a single tree.

When we build decision trees, the root node of each subtree is chosen among attributes which have not yet been chosen by ancestor nodes. The chosen attribute is the best split based on the average goodness of the split in a broad sense. For example, the Gini index [52] in CART, which is one of the most common attribute selection criteria with entropy-based measures, considers the purity of children by the proportion of instances or objects in those children based on the following equation:

$$\text{Gini(C)} = 1 - \sum_j P_j^2$$

where P_j is the probability of class j in node C.

For each possible split the impurity of each split is considered and the one that reduces impurity maximally is chosen. This is similar to the entropy-based attribute selection criteria like gain or gain ratio which also choose an attribute that has the best score on average. (Refer to Section 3.3.)

Therefore, the decision trees built in this way result in average to good prediction models for the target population. In other words, these methods only can find a good prediction for average cases. Depending on the application, we may need very accurate prediction rules possibly through the checking of a few critical feature values.

But a single tree may lead to many unnecessary tests of attributes and may not represent rules that are best for some substantial subset or collection of the objects in a database. Otherwise, the classes of the objects may be decided more accurately by using additional rules of high confidence that are not in the decision tree, and possibly without testing additional attribute values. For example, suppose we have the situation of the following (adapted from Friedman, Gohavi, and Yun [28], p.719).

"A domain requires one to classify patients as sick or healthy. A Boolean attribute denoting whether a person is HIV positive is extremely relevant since we will assume that such persons should be classified as sick. Even though all instances having HIV positive set to true have the same class, a decision tree is unlikely to make the root test based on this attribute because the proportion of these instances is so small; the conditional or average entropy of the two children of a test on the HIV-positive feature will not be much different from the patient and hence the information gain will be small. It is therefore likely that HIV-positive instances will be fragmented throughout the nodes in the tree."

This example presents to us the need of an exhaustive rule finding method that can discover such minor rules with high prediction accuracy. These rules are then attached to decision trees as additional shortcuts for decisions, thus, these rules can strengthen the utility of the decision trees. But since the target database of knowledge discovery is generally very large, unlimited exhaustive rule set generation is almost infeasible and unnecessary due to computational limit and poor supporting number of objects in most found rules. For the minor rules to be reliable and significant, we need a sufficient number of supporting facts in the database. To achieve this we can

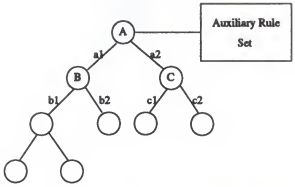


Figure 6.1. A General View of HDT(Hybrid Decision Tree)

use the modified association rule finding algorithm that is presented in the previous chapter with an appropriate minimum support number to find such rules.

6.2 The Generation of HDT

To generate a hybrid decision tree (HDT), we use the modified association rule finding algorithm that is presented in Chapter 5 and 7 to find rules of high prediction accuracy. For the rule set to be of much help, we should select an appropriate minimum support number based on our knowledge of the domain and the size of target database. The highly confident rule set that is found by the modified association rule finding algorithm rather than the decision tree generation algorithms is called an auxiliary rule set. It is attached to a decision tree as shown in Figure 6.1. The procedure to create a HDT is given in Figure 6.2.

Note that the definition of highly confident rules is dependent on the domain and the available data. Usually these are rules of more than 90% confidence. Depending on the application, users may prefer shorter rules with less confidence for rapid or less costly decision making. For example, if each attribute represents a test given to patients to determine the presence or absence of a disease, where each test may require different cost; a physician (and patient) may prefer a smaller less costly decision rule having reasonable confidence to highly expensive precise tests.

1. Run the modified association rule finding algorithm with an appropriate minimum support number.
2. Select the rules of very high/satisfiable confidence.
3. Make a decision tree using your favorite method (C4.5, CART, and so on).
4. Compare the rules found at step 2 and step 3.
5. Make an auxiliary rule set by dropping the rules of step 2 that are included in the decision tree of step 3.

Figure 6.2. The HDT(Hybrid Decision Tree) Generation Procedure

When we generate the decision tree, we can use any decision tree generation algorithm since it is independent of the modified association rule set generation.

When making predictions, we first look at the auxiliary rule set to find the decision of existing shortcut or high confidence. If none are found for the instance, we rely on the decision tree.

6.3 Experimenataion

Several experiments were run using the adult database in UCI machine learning repository which has 32561 objects for training and 16281 objects for testing. We used the training data only to generate an exhaustive rule set, but one may use both sets of data to generate more reliable auxiliary rule set. It was anticipated that since the target database of knowledge discovery was very large, we may generate more reliable rules which are supported by a large number of objects. In this respect, a large population is a help rather than an obstacle in the discovery process.

For the first experiment we supplied the minimum support number of 6200 with confidence threshold of 95% and found eight rules as in Figure 6.3. Here class 1 means yearly income is less than US$50,000. The third rule was also found in the

marital-status = Never-married → class 1 (frq=10191, cf=95.39%)
workclass = Private & marital-status = Never-married
→ class 1 (frq=7858, cf=95.99%)
marital-status = Never-married & capital-gain = 3124
→ class 1 (frq=10168, cf=96.66%)
marital-status = Never-married & capital-loss = 136
→ class 1 (frq=9921, cf=95.67%)
workclass = Private & marital-status = Never-married & capital-gain = 3124
→ class 1 (frq=7838, cf=97.10%)
workclass = Private & marital-status = Never-married & capital-loss = 136
→ class 1 (frq=7659, cf=96.19%)
marital-status = Never-married & capital-gain = 3124 & capital-loss = 136
→ class 1 (frq=9898, cf=96.98%)
workclass = Private & marital-status = Never-married & capital-gain = 3124
& capital-loss = 136 → class 1 (frq=7639, cf=97.34%)

Figure 6.3. A Rule Set Found By the Modified Association Rule Finding Algorithm

decision tree that was made using the C4.5 algorithm as indicated by an astrisk in Figure 6.4, so it may be dropped.

For the second experiment we supplied a minimum support number of 3200 with a confidence threshold of 95% and found 65 rules. Generated were two size 2 rules, thirteen size 3 rules, twenty two size 4 rules, twenty one size 5 rules, and seven size 6 rules. Among them one rule was also duplication in the decision tree. Most of larger rules are refinements of the shorter ones, thus they have a higher confidence than the shorter rules.

6.4 Conclusions

The lazy decision tree algorithm [28] is a nearest neighbor algorithm that finds the most similar cases in a training set with the input instances. It builds a path for each input instance for prediction and has been found to have better prediction accuracy than conventional decision tree algorithms. Because its sole purpose is prediction, it does not generate a model or rule set describing how a decision is made. As a result,

```
C4.5 [release 8] decision tree generator Sun Jun 21 10:37:45 1998
----------------------------------------

    Options:
Pruning confidence level 10%
File stem <adu2>

Read 32561 cases (14 attributes) from adu2.data
          ...
          ...
          ...
Simplified Decision Tree:

capital-gain = 9374: >50K (754.0/70.7)
capital-gain = 15624: >50K (454.0/2.3)
capital-gain = 21874: >50K (38.0/3.9)
capital-gain = 28124: >50K (49.0/2.2)
capital-gain = 34374: <=50K (5.0/1.8)
capital-gain = 40624: <=50K (2.0/1.4)
capital-gain = 96874: >50K (159.0/2.3)
capital-gain = ?: <=50K (0.0)
capital-gain = 3124:
|   marital-status = Divorced: <=50K (4309.0/372.1)
|   marital-status = Never-married: <=50K (10519.0/374.8) *
|   marital-status = Separated: <=50K (1006.0/61.2)
|   marital-status = Widowed: <=50K (967.0/76.2)
|   marital-status = Married-spouse-a: <=50K (408.0/33.6)
|   marital-status = ?: <=50K (0.0)
|   marital-status = Married-civ-spou:
|   |   education-num = 1: <=50K (19.0/2.2)
|   |   education-num = 3: <=50K (249.0/23.4)
                    ......
                    ......
                    ......

|   |   education = 10th: <=50K (0.0)
|   |   education = Doctorate: <=50K (0.0)
|   |   education = 5th-6th: <=50K (0.0)
|   |   education = Preschool: <=50K (0.0)
|   |   education = ?: <=50K (0.0)
```

Figure 6.4. A Decision Tree generated by C4.5

its understandability is limited. Moreover, each prediction will be costly if the target database table is very large.

Rough set theory–based rule finding systems are able to find a minimal size rule set for decision tables of moderate size. The weak points of the systems are scalability and difficulty in dealing with objects in boundary region. One way to use the rough set concept may be to find minimal rules for positive region only on the condition that the positive region is a moderate size.

The HDT approach presented in this chapter has several good points that make it well suited for the task of knowledge discovery in very large databases. First, it is very good for applications that require very reliable decisions because of the extensive examination of the data by the modified association rule finding algorithm. If confident rules are not available in the auxiliary rule set, one may find a rule or a path from the decision tree allowing the HDT to guarantee at least a reasonable decision.

Second is the savings in test cost. If the class of the instance can be predicted by a short rule in the auxiliary rule set, testing cost may be reduced since the evaluation of other attribute values of the instance are not required.

Third, because several efficient association rule finding algorithms are available, for example, large main memory-based algorithm like AprioriTid [2], a hash-table based algorithm [57], random sample based algorithms [58, 74, 75], or even a parallel version of the algorithm [65], and the modified association rule finding algorithm is easily adaptable to the algorithms, our method has the possibility of good scalability.

In the next chapter we present a decision tree induction method based on frequent patterns for better understandability.

CHAPTER 7
DECISION TREE INDUCTION BASED ON FREQUENT PATTERNS

7.1 Introduction

Decision trees have been developed mostly for prediction purposes, so they do not reflect users' interest in the discovered knowledge, and the difficulty in understanding a large generated trees is a well known problem. This point was well indicated by D. Michie([51], p. 233),

"Recent results have shown that programs constructed by systems such as Quinlan's ID3 can be, in one sense, super-programs and at the same time quite incomprehensible to people.... The ID3-synthesised program clearly qualifies as a super-program. Further and perhaps alarmingly, however hard they try, chess experts cannot understand it. Even though it constitutes a complete and correct description, it does not qualify as a concept expression."

To be more comprehensible to the user, it has been found empirically that trees should be as small and/or as linear as possible [4, 52]. The conventional method to generate smaller decision trees relys on pruning. But, because the pruning is based on the principle of minimizing prediction errors, it prunes branches that do not influence error rate much compared to their size. As explained in the previous chapter, because decision trees are average predictors, which means each branch or subtree is just a good approximate for a minimal subtree, it is highly possible the

pruned tree is still not small enough. Moreover, since the tree is built for the whole database, the pruned tree may be too large for human comprehension.

Another concern is the users' interests in the knowledge discovery process. The target databases of knowledge discovery contain many manifest facts and are very large. So, decision trees made without considering users' interests may be so meaninglessly large to hinder understandability.

One way to reflect users interests is to let users choose the root attribute of each subtree. But, as the problem of finding the smallest decision tree is NP–complete problem [39], there is a tremendous number of possible choices for the root attribute of each subtree. Another method to solve this problem may be to show users some smaller size decision trees to make them more understandable and allow the users to refine some of the interesting branches further.

Since more general concepts occur more frequently, if we build a decision tree where more general concepts are located higher in the tree, it will be easy for human to understand the property of each subtree or branch. As a result, the users can determine more easily whether a terminal node should be expanded further or not. The task to place more general concepts higher in the generating decision tree involves the exhaustive rule set generation process based on some minimum support number. Moreover, if we choose the attributes of very confident rules and build decision trees using the attributes that are in these rules, the resulting decision tree may have the tendency to be linear because the corresponding branches to the rules will end early. This will provide users with a better understanding of the generated trees.

Because relational databases are one of the most common databases used currently, in this chapter we show a decision tree induction method for relational databases with the intention to generate more meaningful and understandable decision trees. Basically our method consists of the following three major steps:

Course	Teacher
c_1	t_1
c_2	t_2
c_3	t_1

Course	Student	Grade
c_1	s_1	A
c_1	s_2	A
c_1	s_3	A
c_2	s_1	B
c_2	s_2	B
c_2	s_4	A
c_3	s_1	A
c_3	s_4	A

Figure 7.1. CT and CSG relations

1. Joining the necessary relations focusing on a central relation to make a table,

2. Preprocessing the table, and

3. Generating a comprehensible decision tree.

7.2 Making a Table from Several Relations

Because we are interested in discovering rules for a relational database, we start by creating a table that comprises all of the relations of interest. To make a table, we need to perform a natural join of the related relations, by focusing on a relation that has central meaning for data mining.

If a target database is a simple table like a transaction database or a file, we can apply some pattern finding inductive learning algorithms directly since each tuple or row has a unique meaning. But if the target database is composed of several relations like a relational database, we might need to perform a join operation first since the interesting patterns might be inter–relational. In this case, we need to be careful in applying the pattern finding algorithm since there might be many pseudo patterns due to multi-valued dependencies. For example, suppose we have the CT and CSG relations in Figure 7.1. Joining these two relations results in CTSG relation of Figure 7.2.

Course	Teacher	Student	Grade
c_1	t_1	s_1	A
c_1	t_1	s_2	A
c_1	t_1	s_3	A
c_2	t_2	s_1	B
c_2	t_2	s_2	B
c_2	t_2	s_4	A
c_3	t_1	s_1	A
c_3	t_1	s_4	A

Figure 7.2. CTSG relation

As you see from the CTSG relation in Figure 7.2, there are many pseudo patterns in the new relation. If we are interested in finding frequent patterns from the joined table, we might find c_1t_1, c_2t_2, c_3t_1. But these patterns have no meaning since they can be directly extracted from the CT relation. So, caution is needed when we make a joined table from relational databases. In this example, a good joined table of input for the data mining step results from projecting only on Teacher, Student, and Grade.

7.3 Generating a Comprehensible Decision Tree

7.3.1 The Modified Association Rule Finding Algorithm

The first step of our method begins by finding frequent patterns or rules. To find them, we use the modified association rule finding algorithm to generate an exhaustive rule set supported by a given minimum support number. See the modified association rule finding algorithm (part 1) in Figure 7.3, for details. The procedure generate_candidate is the same as in Chapter 5.

7.3.2 Rule Set Generation

To generate a rule set, we select rules that have more than a given threshold of confidence. The confidence of a rule a1 $\rightarrow$ d1 is

```
F1 = { frequent 1-itemsets}
For k=2 as long as the size of Fk is more than 0 do
      Ck = generate_candidates(Fk-1 )
      For all rows r in the database decision table do
            For all combinations c of size k using r
                  Match c to candidate c' in Ck
                  If matched then
                        Increment counter of c'
                  endif
            endfor
      endfor
Fk = {c' in Ck | the counter of c' ≥ min_support_no}
endfor
```

Figure 7.3. The Modified Association Rule Finding Algorithm (part 1)

$$|a1\ d1| \ / \ |a1|$$

where a1 is an itemset in the condition part and d1 is an itemset in the decision part and | | denotes the frequency of the itemsets. If a rule "A" has the same decision part as another rule "B," but A's conditions are a superset of B's, where A's confidence is less or equal to B's, then "B" should be selected.

7.3.3 Decision Tree Generation Algorithm For Understandability

Figure 7.4 details each step of the algorithm. Note that if we have several candidate decision attributes and a single decision attribute is desired, we can use the decision attribute selection method in Chapter 4.

The reason behind the selection of frequent attributes is that since the rules are exhaustive, if we make decision tree with the attributes in the rules, each subtree has high possibility to stop early. As a result, the possibility of a smaller tree becomes larger. Note that adding most frequently occurring attributes one by one does not guarantee the generation of a skewed tree. So, one may have to be satisfied with the smallest one or most skewed one among generated trees. Note that since our

1. Run the modified association rule finding algorithm with appropriate minimum support number and confidence level.
2. Select an initial attribute set based on the frequency of each attrubute in found rules. (About $1/4 \sim 1/3$ of the attributes based on the property of target data.)
3. Project the database table using the selected attributes.
4. Make a decision tree with the projected table.
5. If a smaller/more skewed tree is needed then
   add attributes based on frequency in the rule set one by one
   until a satisfiable tree is created or try limit.
6. Choose the one with satisfiable size/shape and error rate.
7. Repeat this process to refine interesting branches further.

Figure 7.4. The Decision Tree Generation Algorithm for Understandability

method uses a limited attribute set, the overall error rate in the initial tree may be larger than the conventional method. Further refinement based on users' interests will reduce the error rate.

7.3.4 Time Complexity

The time complexity of the original association rule finding algorithm is O(k log (m/k)) where m is the number of items in the database and k is the size of the frequent itemset [2]. This is considered a reasonable response time. Moreover, since we do not need exact association rules, we can use samples so that we can save significantly on computation. For decision trees, we try only a few decision trees based on the select attributes, and the cost at each node of tree is $O(|r||A|)$ where $|A|$ is the number of attributes. It has been known that the induction does not exponentially grow in time and space [62].

Table 7.1. Attributes and Their Frequency in Rules for Adult Database

attribute number	attribute name	frequency
1	age	14
2	workclass	25
3	fnlwgt	0
4	education	0
5	education-num	4
6	marital-status	47
7	occupation	0
8	relationship	30
9	race	2
10	sex	9
11	capital-gain	33
12	capital-loss	32
13	hours-per-week	0
14	native-country	17

7.4 Experimentation

We used a real world data set, a semi-real world data set, and a random data generation program from the UCI machine learning data repository [54]. As a real world data set, a database called adult which contains 32,561 objects for training was chosen.

By using a minimum support number of 3200 and a confidence level of 95%, we found 65 rules. The frequency of each attributes in the rule set is in Table 7.1.

Based on the frequency of attributes, we selected four attributes, marital-status, relationship, capital-gain, and capital-loss as the initial attribute set. These were used to build a decision tree using C4.5 algorithm. The resulting tree has 79 nodes with an estimated error rate of 19.1% when the pruning confidence level is 25%. The complexity of the tried trees and their expected error rates are shown in Table 7.2. Among them, the third tree may be chosen because it is the smallest and most skewed.

Table 7.2. The Summary of Tried Decision Trees Based on The Frequency of Attributes for Adult Database

used attribute number	tree size	estimated error rate
6, 8, 11, 12	79	19.1%
2, 6, 8, 11, 12	95	18.6%
2, 6, 8, 11, 12, 14	50 *	18.6%
1, 2, 6, 8, 11, 12, 14	94	18.5%
1, 2, 6, 8, 10, 11, 12, 14	71	18.5%
1, 2, 5, 6, 8, 10, 11, 12, 14	191 **	15.9%

Table 7.3. The Summary of Decision Trees Based on Different Levels of Pruning Confidence By C4.5 for Adult Database

tree size	estimated error rate	pruning confidence
569	15.5%	25%(default)
368 **	16.1%	15%
138	16.8%	5%
81	17.6%	0.5%
81 *	18.7%	0.01%
75	18.8%	0.001%

This tree is shown in Figure 7.5. In the decision tree, terminal nodes that do not cover any objects are dropped for simplicity.

For comparison with the pruning based simple tree generation method, we generated decision trees using all attributes for several levels of pruning confidence. Note that the lower the pruning confidence is, the severer the pruning is. The low limit of the confidence is 0.001% in C4.5. The result is shown in Table 7.3. As indicated by * and ** in the table, our method generated decision trees of $2/3 \sim 1/2$ of the size of the tree with similar expected error rates so that the understandability of the generated tree becomes greater.

```
Simplified Decision Tree:

capital-gain = 9374: >50K (754.0/65.9)
capital-gain = 15624: >50K (454.0/1.4)
capital-gain = 21874: >50K (38.0/2.6)
capital-gain = 28124: >50K (49.0/1.4)
capital-gain = 34374: <=50K (5.0/1.2)
capital-gain = 40624: <=50K (2.0/1.0)
capital-gain = 96874: >50K (159.0/1.4)
capital-gain = 3124:
|   marital-status = Divorced: <=50K (4309.0/361.1)
|   marital-status = Never-married: <=50K (10519.0/363.5)
|   marital-status = Separated: <=50K (1006.0/56.6)
|   marital-status = Widowed: <=50K (967.0/71.1)
|   marital-status = Married-spouse-a: <=50K (408.0/30.2)
|   marital-status = Married-AF-spous: <=50K (22.0/11.1)
|   marital-status = Married-civ-spou:
|   |   capital-loss = 408: <=50K (3.0/1.1)
|   |   capital-loss = 1225: <=50K (4.0/1.2)
|   |   capital-loss = 1497: <=50K (94.0/34.7)
|   |   capital-loss = 1769: >50K (506.0/109.9)
|   |   capital-loss = 2041: >50K (238.0/68.3)
|   |   capital-loss = 2314: >50K (103.0/32.8)
|   |   capital-loss = 2586: <=50K (10.0/1.3)
|   |   capital-loss = 136:
|   |   |   workclass = Private: <=50K (8892.1/3281.5)
|   |   |   workclass = Self-emp-not-inc: <=50K (1516.9/440.7)
|   |   |   workclass = Self-emp-inc: >50K (642.8/289.7)
|   |   |   workclass = Federal-gov: >50K (411.4/179.8)
|   |   |   workclass = State-gov: <=50K (526.6/239.6)
|   |   |   workclass = Without-pay: <=50K (8.4/1.4)
|   |   |   workclass = Never-worked: <=50K (1.0/0.8)
|   |   |   workclass = Local-gov:
|   |   |   |   relationship = Wife: >50K (134.3/56.6)
|   |   |   |   relationship = Own-child: <=50K (1.6/0.9)
|   |   |   |   relationship = Husband: <=50K (767.4/333.8)
|   |   |   |   relationship = Not-in-family: <=50K (0.2/0.2)
|   |   |   |   relationship = Other-relative: <=50K (8.3/1.3)
```

Figure 7.5. A Skewed Decision Tree of Attribute-Selected Adult Database

Table 7.4. Attributes and Their Frequency in Rules for Nursery Database

attribute number	attribute name	frequency
1	parents	2
2	has-nurs	5
3	form	3
4	children	3
5	housing	2
6	finance	1
7	social	2
8	health	19

The next step may be further refinement of interesting terminal nodes with relatively higher error rates. Similar work can be done after selecting objects that belong to the terminal nodes.

The second experiment was done with a semi-real world database called Nursery. Nursery database was derived artificially from a hierarchical decision model originally developed to rank applications for nursery schools. It was used during several years in 1980's when there was excessive enrollment to these schools in Ljubljana, Slovenia for objective explanation for rejected applications. So, the property of the data has a tendency of uniformity in the distribution of condition and decision values. It has eight attributes, parents, has_nurs, (application) form, children, housing, finance, social, health, and the class. All attributes have nominal values. The number of records of database is 12960.

By using minimum support number of 600 and confidence level of 95%, we found 19 rules. The reason why the minimum support number is about 5% of the total data is that it has versatile values because of its semi-artificial nature. The frequency of each attributes in the rule set is in Table 7.4.

Based on the frequency of attributes, we selected attribute health, has-nurs, and form for the initial attribute set, and made a decision tree using C4.5. The resulting

Table 7.5. The Summary of Tried Decision Trees Based on The Frequency of Attributes for Nursery Database

used attribute number	tree size	estimated error rate
2, 3, 8	14	17.5%
2, 3, 4, 8	46	16.9%
1, 2, 3, 4, 8	88 *	10.7%
1, 2, 3, 4, 5, 8	119	9.9%

Table 7.6. The Summary of Decision Trees Based on Different Levels of Pruning Confidence By C4.5 for Nursery Database

tree size	estimated error rate	pruning confidence
511	5.5%	25%(default)
487	6.7%	15%
327	8.7%	5%
151 *	11.5%	0.5%
63	14.3%	0.01%
63	14.7%	0.001%

tree has 14 nodes with estimated error rate of 17.5% when the pruning confidence level is 25%. The complexity of the tried trees and their expected error rates are shown in Table 7.5.

For comparison with the pruning based tree simpification method, we experimented for several levels of pruning confidence using C4.5 algorithm. The result is shown in Table 7.6. As in the table, our method generated a decision tree of about 1/2 of the size of the tree with a similar expected error rate as indicated by *.

For the third experiment, we used the random data sets generated by the dgp2 program, which was used for the experiment in Chapter 4. Because the data distribution is in variety, the minimum support number of 7500 was given. The data has 100,000 records. The frequency of each attribute in the rule sets with confidence of more than 95% for each data set is shown in Table 7.7. In the table frequecy 1 is for

Table 7.7. Attributes and Their Frequency in Rules for Random Data

attribute number	attribute name	frequency 1	frequency 2	frequency 3
1	A	0	0	0
2	B	19	18	19
3	C	18	17	18
4	D	23	18	25
5	E	22	15	24
6	F	20	18	24
7	G	14	13	14
8	H	17	16	18
9	I	15	13	16
10	J	43	27	44
11	K	6	5	6
12	L	22	20	27
13	M	16	14	16
14	N	15	15	19
15	O	21	19	22
16	P	17	17	18
17	Q	22	19	16
18	R	19	17	22
19	S	17	16	17

the data of no conflicts (from 172 rules), frquency 2 is for the data of 20% conflicts (from 149 rules), and frequency 3 is for the data of 10% conflicts (from 186 rules) in decisions.

The summary of generated decision tree for the random data set of no conflict decisions, 20% conflicts, and 10% conflicts are shown in Table 7.8, Table 7.10, and Table 7.12 respectively. The summary of the generated trees simplified by pruning only is shown in Table 7.9 for the no conflict data set, Table 7.11 for 20% conflicts, and Table 7.13 for 10% conflicts for comparison.

In Table 7.8 the last two rows were generated to see the effect of attribute selection. Comparing the first two rows in Table 7.9, we found that only six or seven attributes among nineteen attributes has major effect for the decisions. Moreover, we found

Table 7.8. The Summary of Tried Decision Trees Based on The Frequency of Attributes for The Random Data Set of No Conflict Decisions

used attribute number	tree size	estimated error rate
4, 10	78 *	17.6%
4, 10, 12	468 **	13.1%
4, 5, 10, 12	1174	11.3%
4, 5, 10, 12, 17	3315	10.0%
4, 5, 10, 12, 15, 17	7777	9.3%
4, 5, 6, 10, 12, 15, 17	10941	8.7 %

Table 7.9. The Summary of Decision Trees Based on Different Levels of Pruning Confidence By C4.5 for The Random Data Set of No Conflict Decisions

tree size	estimated error rate	pruning confidence
14458	7.7%	25%(default)
9752	9.2%	15%
5069	11.3%	5%
2477 **	13.1%	0.95%
1461	13.5%	0.5%
471	15.6%	0.01%
377 *	15.8%	0.001%

that our method generated trees of $1/5 \sim 1/4$ size compared to the other method when the estimated error rate is similar. See the starred rows for comparison. Similar results were obtained from the random data of 20% conflicts and 10% conflicts. See the starred rows in the corresponding tables.

7.5 Sampling for Very Large Databases

If a target database is very large, the joined table might be very large, so usually the joined table is implemented in a materialized view for efficiency [13]. There is an exact association rule finding algorithm based on random samples [74, 75] with the automatic detection of the possibility of missed frequent itemsets. The sufficient sample sizes are summarized in Table 7.14 (adapted from Agrawal et al. [2]). The

Table 7.10. The Summary of Tried Decision Trees Based on The Frequency of Attributes for The Random Data Set of 20% Conflict Decisions

used attribute number	tree size	estimated error rate
10, 12	78 *	21.0%
10, 12, 17	325 **	18.8%
10, 12, 15, 17	1030	18.1%
6, 10, 12, 15, 17	2116	17.6%
4, 6, 10, 12, 15, 17	3037	17.2%
2, 4, 6, 10, 12, 15, 17	3806	17.2 %

Table 7.11. The Summary of Decision Trees Based on Different Levels of Pruning Confidence By C4.5 for The Random Data Set of 20% Conflict Decisions

tree size	estimated error rate	pruning confidence
5857	16.7%	25%(default)
3015	17.9%	15%
1679 **	18.7%	9%
1440 **	18.9%	8%
1244	19.4%	5%
373 *	20.9%	0.5%
195	22.9%	0.01%
195	23.0%	0.001%

Table 7.12. The Summary of Tried Decision Trees Based on The Frequency of Attributes for The Random Data Set of 10% Conflict Decisions

used attribute number	tree size	estimated error rate
10, 12	78 *	18.8%
10, 12, 17	364 **	15.7%
6, 10, 12, 17	1157	14.3%
5, 6, 10, 12, 17	2653	13.8%
4, 5, 6, 10, 12, 17	5168	13.2%
4, 5, 6, 10, 12, 15, 17	10941	12.9 %

Table 7.13. The Summary of Decision Trees Based on Different Levels of Pruning Confidence By C4.5 for The Random Data Set of 10% Conflict Decisions

tree size	estimated error rate	pruning confidence
9875	12.3%	25%(default)
5586	13.7%	15%
2451	15.4%	5%
2272 *	15.7%	4.2%
620	17.2%	0.5%
418 *	18.8%	0.04%
392	19.2%	0.01%
380	19.3%	0.001%

Table 7.14. Sufficient Sample Sizes, Given Values for α and Probabilities of Error More Than α

$\alpha =$	1%	0.1%	0.01%	0.001%
$P(error > \alpha) \approx 1\%$	23000	2.3×10^6	2.3×10^8	2.3×10^{10}
$P(error > \alpha) \approx 5\%$	15000	1.5×10^6	1.5×10^8	1.5×10^{10}
$P(error > \alpha) \approx 10\%$	11500	1.15×10^6	1.15×10^8	1.15×10^{10}

algorithm needs to scan the whole database to find missed frequent itemsets at the second pass. Note that such probability is very low if we have sufficient samples.

If users are satisfied with an approximate association rule set, more computation time can be saved by empirically setting sample size and not going back one more pass to remedy sampling errors [58].

If a materialized view is unavailable, the following method may save space and time.

Definition. Let R be a relation, a_i be the ith value of attribute A and A is the join attribute, then

$$max|R_{a_i}|$$

is the maximum cardinality in relation R of any join domain value a_i for all i.

The random sample from the natural join of two relations without generating the whole joined table can be made by the algorithm in Figure 7.6 [56]. The process of the algorithm can be illustrated by the following example.

Example. Suppose we have picked a tuple of relation R randomly, and join it with tuples in relation S as in Figure 7.7. In the figure, because each value of attribute A occurs different number of times, it is not fair if we take all the tuples. If we take a tuple of relation S that has joined for value a1, the sampling become biased to relation R. So, to be fair also to relation S, it is accepted with a probability proportional to the relative size of T. The concept is illustrated in Figure 7.8 where we joined two relations with all values of attribute A. In the figure, a2 has maximum cardinality. So, if a sampled tuple has a value of a2, its acceptance probability is 1. Sampling from multiple joins are just repeated sampling operations of the sampling from the join of two relations.

- Input: relations R and S.
- Output: a random sample of size n of natural join.

Repeat

1. Sample a tuple from R.
2. Join the tuple to S, yielding T.
3. Sample a tuple from T.
4. Accept the tuple with probability $\frac{|S_{a_i}|}{max|S_{a_i}|}$

Until a sample size n is obtained.

Figure 7.6. Random Sampling for Natural Join

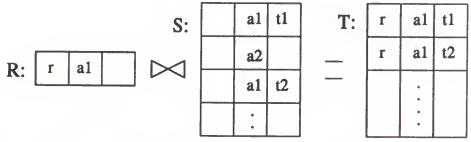


Figure 7.7. Natural join of a Tuple in Relation R and Relation S

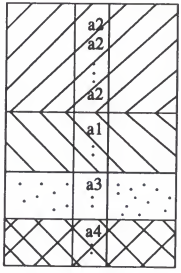


Figure 7.8. The Conceptual Diagram of The Reason for Acceptance Test

Because our algorithm can be adapted easily to the sampling-based algorithms and the attribute selection criteria does not need exact association rules, randomized samples can reduce significant amount of computation time and space for very large databases.

7.6 Conclusions

Since most decision trees have been developed for predictional purposes, they have two well known problems. The discovered knowledge does not reflect the domain knowledge and the generated trees are difficult to understand because they are large. To be more meaningful and understandable decision trees, the trees should be as small and as linear as possible with the reflection of the domain knowledge to the generated decision trees.

To reflect the users' interests and build an understandable tree, we give the users a chance to select a good one among small number of possible trees. Note that building the smallest tree by selecting attribute set based on the combination of each attribute requires exponential number of trials. So, as a heuristic to build a small tree, we use the information on frequency of attributes in frequent patterns/rules. Further refinement of the terminal nodes with relatively high error rates can be done in the same manner.

A good feature of our method is the generated tree may not only become smaller but also have higher confidence by focusing on the attributes of frequent rules with high confidence. Moreover, the generated tree has the capability of comprising users' interests by allowing them to choose branches for further refinement. Note that this interactive work is only possible when the tree size is within human comprehensibility.

In the next chapter we conclude our research with future directions.

CHAPTER 8
CONCLUSIONS AND FUTURE RESEARCH

8.1 Conclusions

Several problems exist for data mining in real world databases: the difficulty of determining a decision attribute when limited domain knowledge exists, or the difficulty in selecting a decision attribute from a new table formed from joining several relations, the problem of reflecting users' interests in the knowledge discovery process, the prblem of finding a better prediction model for very large databases, and the understandability problem of large decision trees from very large target databases.

The important assumption of classification systems like decision tree systems is a decison attribute is predefined. But, this assumption may be a limitation to the utility of the systems. Because, in general, the target databases of KDD is not made with the intention to discover some useful information so that they are not organized for the discovery. As a result, users may not be sure which attribute in a database table is a good decision attribute among possible candidates due to limited domain knowledge. As a method to select a better decision attribute, we developed an objective assessing method which considers the consistency of the data and simplicity of rule sets based on two well-known techniques; rough set theory and decision tree generation method. Depending on the need of the application and the property of the data, one can tune between more precise rule set and smaller rule set using the method.

Because the target databases of KDD are very large and contain many manifest facts, the direct application of conventional machine learning algorithms to databases may not generate a useful results. The rule sets generated by the decision tree

methods are only a small part in the vast possibilities that can be generated and it is difficult to fully incorporate the users' interests in generating decision trees. As a consequence, the resulting trees may not be as useful as expected. The genetic algorithm-based learning systems have the difficulty of devising a "good" fitness function that can reflect users' interests well since, in general, the users' interests or domain knowledge is not well defined. Moreover, these systems require intensive computation for training. By taking advantage of the fact that more general concepts occur more frequently, our stepwise refinement technique considers all possibilities in generating rules with respect to the given minimum support number, so it can generate an exhaustive rule set in the direction of the users' interests at each level of the given minimum support number. So, the stepwise refining of rule sets, which is based on the frequency of data, can allow users to find the best rule set of interest by helping them to use their ill-defined domain knowledge. So, this process avoides the possibility of generating numerous rules. Another good point of our method is that the number of supporting objects for the generated descriptive rules is large enough to be considered statistically significant because of the large size of the target databases. This allows these rules to have good prediction accuracy also. The weak point of our method is its high computational complexity and its comprehensibility problem if the minimum support number in the early steps is small.

Because conventional decision trees are average to good prediction models, they may not contain very accurate branches or rules that are best for some substantial collection of objects in a database, and they may lead to unnecessary tests of attribute values due to fragmentation effects. To surmount these weak points, our HDT (hybrid decision tree) can be used as a good alternative. The HDT has a conventional decision tree and an exhaustive rule set, called auxiliary rule set, which is supported by a large number of objects. Because the auxiliary rule set has rules that are best and minimal

for some substantial collection of objects in a database, it can be used not only to save testing cost for some attribute values of testing instances but also to make a very reliable decision. Moreover, if confident rules are not available in the auxiliary rule set, one may find a rule or path from the decision tree allowing the HDT to guarantee at least a reasonable decision.

In general, the target databases of KDD may have many manifest facts and may be large. As a result, the direct application of the decision tree generation methods may generate very large trees with lots of meaningless subtrees or branches, thus the trees are not as useful as expected. As a way to build an understandable tree and reflect users' interests, we show users smaller trees of confidence for better understandability, and allow users to refine branches of their interest to generate a more meaningful tree. Conventional methods to build small decision trees rely on pruning only. But, because the target database may be very large in KDD domain, the resulting tree may not be small enough for human comprehension. For better human comprehension, the tree is built based on the attributes that occur frequently in an exhaustive rule set of high confidence. The rules are also supported by a large number of objects bounded by a given minimum support number. Because the related subtrees have high possibility of stopping growing early, it may generate smaller trees. Further refinement of the terminal nodes of interest can be done in the same manner. That is, by giving the system different minimum support numbers, stepwise refinement is possible. So, users can grasp a higher view of the target database by using a larger support number, then they can focus on more specific parts of the generated decision tree by making a table which satisfies a specific condition. For the case of multiple candidate decision attributes, users may use our objective assessment method to determine which one is best for their purpose. Because a large portion of real world

databases are relational databases, we assumed the target databases are the joined tables of relational databases.

Some research on faster decision tree generation and association rule finding algorithms has been done, so faster turnaround time will be also applicable to our method.

8.2 Future Research

There are three issues that will enhance the capability of our system; employing a good graphic user interface systems, fuzzy numbers for numeric values, and knowledge–based inference systems.

In this research, the main concern was how to generate a simpler rule set that accurately reflects the users' interests well. All experimentation were done using a prototype system that is not convenient to general users. A good graphic user interface will enhance the use of this approach by making it convenient for users to select interesting/uninteresting rules and will make these methods available to a broader group of users.

As indicated in chapter 4, converting numeric values into nominal values is prone to be subjective. In this respect, employing fuzzy numbers [43, 80] in the conversion task will increase the number of found rules and their supporting number of objects. So, it may allow the method to find rules that have not been found by conventional means since the employment of fuzzy numbers will increase the case of each individual nominal value. Note that the adoption of fuzzy numbers here is the reverse of the conventional usage of the numbers. While conventional fuzzy numbers try to convert imprecise concepts or expressions into intervals of real numbers, we want to convert intervals which may be intersected with others into nominal values.

Since our system relies on frequency only to make a comprehensible rule set size from which users select interesting/uninteresting rules, our method may not be

suited for some domains. In these domains if the generated rule set is very large, it would be better to rely on some built–in knowledge base and inference engine [30] to select the interesting/uninteresting rules. Research on active/objective–oriented databases [13, 69], which allows the seamless integration of database and knowledge–base technologies, might shed additional light on knowledge discovery in databases.

REFERENCES

[1] R. Agrawal, T. Imielinski, and A. Swami. Mining Association Rules between Sets of Items in Large Databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data:SIGMOD-93*, pages 207–216, Washington, D.C., May 26–28 1993.

[2] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A.I. Verkamo. Fast Discovery of Association Rules. In U.M. Fayyad, G. Piatetsky-Shapiro, P. Smith, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 307–328. AAAI Press/The MIT Press, 1996.

[3] S. Angier, G. Venturini, and Y. Kodratoff. Learning First Order Logic Rules with A Genetic Algorithm. In U.M. Fayyad and R. Uthurusamy, editors, *Proceedings: The First International Conference on Knowledge Discovery and Data Mining:KDD'95*, pages 21–26, Montreal Quebec, Canada, Aug. 20 – 21 1995.

[4] B. Arbab and D. Michie. Generating Expert Rules from Examples in Prolog. In J.E. Hayes, D. Michie, and J. Richards, editors, *Machine Intelligence 11*, pages 289–304. Oxford University Press, Oxford, UK, 1988.

[5] L. Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone. *Classification and Regression Trees*. Wadsworth International Group, Inc., 1984.

[6] J. Catlett. *Megainduction: Machine Learning on Very Large Databases*. PhD thesis, University of Sydney, Australia, 1991.

[7] T. Chang and E. Sciore. A Universal Relation Data Model with Semantic Abstractions. *IEEE Transactions on Knowledge and Data Engineering*, 4(1):23–33, 1992.

[8] E. Charniak. Bayesian Networks without Tears. *AI Magazine*, Winter:50–63, 1991.

[9] P. Cheeseman and J. Stutz. Bayesian Classification (Autoclass): Theory and Results. In U.M. Fayyad, G. Piatetsky-Shapiro, P. Smith, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 1–34. AAAI Press/The MIT Press, 1996.

[10] Y. Cheng and K.S. Fu. Conceptual Clustering in Knowledge Organization. *IEEE Transactions on Pattern Analysis and Machine Learning*, PAMI-7(5):592–598, 1995.

[11] D.W. Cheun, J. Han, V.T. Ng, A.W. Fu, and Y. Fu. A Fast Distributed Algorithm for Mining Association Rules. In *PDIS'96*, 1996. http://fas.sfu.ca/cs/han/kdd/FDM96.ps.

[12] T.M. Cover and P.E. Hart. Nearest Neighbor Pattern Classification. *IEEE Transactions on Information Theory*, 13:21–27.

[13] C.J. Date. *An Introduction To Database Systems.* Addison Wesley Publishing Company, sixth edition, 1996.

[14] R. L. de Mántaras. *Approximate Reasoning Models.* Ellis Horwood Limited, 1990.

[15] K.M. Decker and S. Focardi. Technology Overview: A Report on Data Mining. Technical report, Swiss Scientific Computing Center, 1995. Technical Report CSCS TR–95–02.

[16] K.A. DeJong, W.M. Spears, and D.F. Gordon. Using Genetic Algorithms for Concept Learning. *Machine Learning*, 13:161–188, 1993.

[17] R.O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis.* John Wiley and Sons, 1993.

[18] R. Elmasri and S.B. Navathe. *Fundamentals of Database Systems.* The Benjamin/Cummings Publishing Company, Inc., 1989.

[19] Ø.T. Aasheim and H.G. Solheim. Rough Sets as a Framework for Data Mining. Technical report, Project report, Knowledge Systems Group, Faculty of Computer Systems and Telematics, The Norwegian University of Science and Technology, April 30 1996.

[20] F. Esposito, D. Malerba, and G. Semeraro. A Comparative Analysis of Methods for Pruning Decision Trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(5):476–491, May 1997.

[21] U. Fayyad. Data Mining and Knowledge Discovery: Making Sense out of Data. *IEEE Expert*, pages 20–25, October 1996.

[22] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From Data Mining to Knowledge Discovery: An Overview. In U.M. Fayyad, G. Piatetsky-Shapiro, P. Smith, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 1–34. AAAI Press/The MIT Press, 1996.

[23] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From Data Mining to Knowledge Discovery in Databases. *AI Magazine*, Fall:37–54, 1996.

[24] U.M. Fayyad, G. Piatetsky-Shapiro, P. Smith, and R. Uthurusamy, editors. *Advances in Knowledge Discovery and Data Mining.* AAAI Press/The MIT Press, 1996.

[25] D.H. Fisher. Knowledge Acquisition via Incremental Conceptual Clustering. *Machine Learning*, 2:139–172, 1987.

[26] D.H. Fisher. Optimization and Simplification of Hierarchical Clustering. In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining*, pages 118–123, Menlo Park, CA, 1995. AAAI Press.

[27] D.H. Fisher. Iterative Optimization and Simplification of Hierarchical Clustering. *Journal of Artificial Intelligence Research*, 4:147–179, 1996.

[28] J.H. Friedman, R. Kohavi, and Y. Yun. Lazy Decision Trees. In *AAAI-96*, pages 717–724, 1996.

[29] L.M. Fu. *Neural Networks in Computer Intelligence.* McGraw Hill, Inc., New York, 1994.

[30] A.J. Gonzalez and D.D. Dankel II. *The Engineering of Knowledge-based Systems: Theory and Practice.* Prentice Hall, 1993.

[31] J. Han and Y. Fu. Discovery of Multiple-level Association Rules from Large Databases. In *Proceedings of the 21st International Conference on Very Large Databases:VLDB'95*, pages 420–431, Zurich, Swizerland, 1995.

[32] J. Han and Y. Fu. Attribute-Oriented Induction in Data Mining. In U.M. Fayyad, G. Piatetsky-Shapiro, P. Smith, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 339–421. AAAI Press/The MIT Press, 1996.

[33] D. Heckerman. A Tutorial on Learning with Bayesian Networks. Technical report, Microsoft Corporation, 1995. Technical report MSR–TR–95–06.

[34] M. Holsheimer, M. Kersten, H. Mannila, and H. Toivonen. A Perspective on Databases and Data Mining. In *KDD'95*, pages 150–155, Montreal, Canada, August 1995.

[35] M. Holsheimer and A. Siebes. Data Mining: The Search for Knowledge in Databases. Technical report, CWI, 1994. Technical report CS-R9406.

[36] J. Hong and C. Mao. Incremental Discovery of Rules and Structure by Hierarchical and Parallel Clustering. In *Knowledge Discovery from Databases*, pages 177–194. The AAAI Press, 1991.

[37] M. Houtsma and A. Swami. Set-Oriented Mining of Association Rules. In *Proceedings of the Eleventh International Conference on Data Engineering*, pages 25–33, Taipei, Taiwan, March 6–10 1995. IEEE Computer Society.

[38] X. Hu. *Knowledge Discovery in Databases: An Attribute-Oriented Rough Set Approach.* PhD thesis, University of Regina, 1995.

[39] L. Hyafil and R.L. Rivest. Constructing Optimal Binary Decision Trees is NP-Complete. *Information Processing Letters*, 5(1):15–17, 1976.

[40] R.E. Stepages III and R. Michalski. Conceptual Clustering: Inventing Goal–oriented Classification of Structured Objects. In J.G. Carbonell R.S. Michalski and T.M. Mitchell, editors, *Machine Learning Volume 2: An Artificial Intelligence Approach*, pages 471–498. Morgan Kaufmann Publishers, Inc., 1986.

[41] C. Z. Janikow. *Inductive Learning of Decision Rules from Attribute-based Examples: A Knowledge Intensive Genetic Algorithm Approach.* PhD thesis, University of North Carolina, 1991.

[42] M. Klemettinen, H. Mannila, P. Ronkainen, H. Toivonen, and I. Verkamo. Finding Interesting Rules from Large Set of Discovered Association Rules. In *Proceedings of the Third International Conference on Information and Knowledge Management:CIKM'94*, pages 401–407, Gaithersburg, Maryland, 1994. ACM Press.

[43] G.J. Klir and B. Yuan. *Fuzzy Sets and Fuzzy Logic: Theory and Applications.* Prentice Hall, New Jersey, 1995.

[44] E. Krusińska, A. Babic, R. Slowiński, and J. Stefanowski. Comparison of Rough Sets Approach and Probabilistic Data Analysis Techniques on a Common Set of Medical Data. In Roman Slowinski, editor, *Intelligent Decision Support: Handbook of Applications and Advances of the Rough Set Theory*, pages 251–265. Kluwer Academic Publishers, 1992.

[45] B. Lent, A. Swami, and J. Widom. Clustering Association Rules. In *IEEE'97 Proceedings of Data Engineering*, pages 220–231, 1997.

[46] D. Maier, J. Ullman, and M.Y. Vardi. On the Foundation of the Universal Relation Model. *ACM Transactions of Database Systems*, 9(2), 1984.

[47] H. Mannila and H. Toivonen. Multiple Uses of Frequent Sets and Condensed Representations—Extended Abstract. In *2nd International Conference on Konwledge Discovery and Data Mining (KDD'96)*, pages 189–194, Portland, Oregon, August 1996. AAAI Press.

[48] R.A. McCallum and K.A. Spackman. Using Genetic Algorithms to Learn Disjunctive Rules from examples. In *Proceedings of the Seventh International Conference on Machine Learning*, University of Texas, Austin Texas, June 21 – 23 1990. Morgan Kaufmann Publishers, Inc.

[49] M. Mehta, R. Agrawal, and J. Rissanen. SLIQ: A Fast Scalable Classifier for Data Mining. In *Proceedings of EBDT France*, March 1996.

[50] R.S. Michalski, I. Mozetic, J. Hong, and N. Lavrac. The Multi–purpose Incremental Learning System AQ15 and Its Testing Application to Three Medical Domains. In *AAAI*, pages 1041–1045, Philadelphia, 1986.

[51] D. Michie. *On Machine Intelligence.* Ellis Horwood, Chichester,UK, 2nd edition, 1986.

[52] D. Michie, D.J. Spiegelhalter, and C.C. Taylor, editors. *Machine Learning, Neural and Statistical Classification.* Ellis Horwood Limited, 1994.

[53] A. Mrózek. Rough Sets in Computer Implementation of Rule-based Control of Industrial Processes. In Roman Slowinski, editor, *Intelligent Decision Support: Handbook of Applications and Advances of the Rough Set Theory*, pages 19–31. Kluwer Academic Publishers, 1992.

[54] P.M. Murphy and D.W. Aha. UCI Repository of Machine Learning Databases. Technical report, University of California at Irvine, CA, 1994. http://www.ics.uci.edu/~mlearn/mlrepository.html.

[55] Sreerama K. Murthy. *Automatic Construction of Decision Trees from Data: A Multi-Disciplinary Survey.* http://www.cs.jhu.edu/~murthy/survey-paper.ps, 1997.

[56] F. Olken. *Random Sampling from Databases.* PhD thesis, University of California at Berkeley, 1993.

[57] J.S. Park, M. Chen, and P.S. Yu. Using a Hash-Based Method with Transaction Trimming for Mining Association Rules. *IEEE Transactions on Knowledge and Data Engineering*, 9(5):813–825, 1997.

[58] J.S. Park, P.S. Yu, and M.S. Chen. Mining Association Rules with Adjustable Accuracy. In *Proceedings of CIKM-97*, pages 151–160, Las Vegas, Nevada, U.S.A., 1997.

[59] D.W. Patterson. *Introduction to Artificial Intelligence and Expert Systems.* Prentice-Hall, Inc., 1990.

[60] Z. Pawlak. *Rough Sets: Theoretical Aspects of Reasoning About Data.* Kluwer, Netherlands, 1991.

[61] V. Pimat, I. Kononenko, T. Janc, and I. Bratko. Medical Estimation of Automatically Induced Decision Rules. In *Proceedings of the Second European Conference on Artificial Intelligence in Medicine*, pages 24–36, 1989.

[62] J.R. Quinlan. Induction of Decision Trees. *Machine Learning*, 1:81–106, 1986.

[63] J.R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann Publishers, Inc., 1993.

[64] S. Russel and P. Norvig. *Artificial Intelligence: a Modern Approach.* Prentice Hall, 1995.

[65] A. Savasere, E. Omiecinski, and S. Navathe. An Efficient Algorithm for Mining Association Rules in Large Databases. Technical report, College of Computing, Georgia Institute of Technology, 1995. Technical Report No.: GIT–CC–95–04.

[66] R. Słowinski, editor. *Intelligent Decision Support: Handbook of Applications and Advances of the Rough Set Theory.* Kluwer Academic Publishers, 1992.

[67] R. Srikant and R. Agrawal. Mining Generalized Association Rules. In *Proceedings of the 21st International Conference on Very Large Databases:VLDB'95*, pages 407–419, Zurich, Swizerland, 1995.

[68] R. Srikant and R. Agrawal. Mining Quantitative Association Rules in Large Relational Tables. In *SIGMOD'96*, 1996.

[69] M. Stonebraker, editor. *Readings in Database Systems.* Morgan Kaufmann Publishers, San Francisco, California, second edition, 1994.

[70] H. Sug and D.D. Dankel II. Rule Discovery using Patterns from Joined Table of Relational Databases. In *Proceedings of Workshop on High Performance Data Mining held in conjunction with IPPS/SPDP'98*, Orlando, Florida, March 1998. http://www.cise.ufl.edu/~ranka/hpdmp.html.

[71] H. Sug and D.D. Dankel II. Stepwise Rule Set Discovery based on Frequency. In *Proceedings Volume 2: World Multiconference on Systemics, Cybernetics and Informatics:SCI'98*, pages 285–291, Orlando, FL, July 1998.

[72] M. Tan. Cost-sensitive Learning of Classification Knowledge and its Application in Robotics. *Machine Learning*, 13(1):7–33, 1993.

[73] J. Teghem and M. Benjelloun. Some Experiments to Compare Rough Sets Theory and Ordinal Statistical Methods. In Roman Slowinski, editor, *Intelligent Decision Support: Handbook of Applications and Advances of the Rough Set Theory*, pages 267–286. Kluwer Academic Publishers, 1992.

[74] H. Toivonen. *Discovery of Frequent Patterns in Large Data Collections.* PhD thesis, University of Helsinki, Finland, 1996.

[75] H. Toivonen. Sampling Large Databases for Association Rules. In *Proceedings of the 22th International Conference on Very Large Databases:VLDB'96*, pages 134–145, Mumbai(Bombay), India, September 3 – 6 1996. Morgan Kaufman.

[76] H. Toivonen, M. Klemettinen, H. Mannila, P. Ronkainen, and K. Hàtónen. Pruning and Grouping of Discovered Association Rules. In *Workshop Notes of the ECML–95 Workshop on Statistics, Machine Learning and Knowledge Discovery in Databases*, pages 47–52, Heraklion, Crete, Greece, April 1995.

[77] J.D. Ullman. *Principles Database Systems.* Computer Science Press, Inc., 2nd edition, 1982.

[78] J.R.A. Venegas. *On Decision Tree Induction for Knowledge Discovery in Very Large Databases.* PhD thesis, University of Florida, 1996.

[79] R.R. Yager, J. Kacprzyk, and M. Fedrizzi. From Rough Set Theory to Evidence Theory. In *Advances in Dempster-Shafer Theory of Evidence*, pages 193–236. John Wiley and Sons, Inc., 1994.

[80] H.J. Zimmermann. *Fuzzy Set Theory and Its Applications.* Kluwer–Nijhoff Publishing, MA, 1985.

BIOGRAPHICAL SKETCH

Dr. Hyontai Sug received the BS degree from Pusan National University, Republic of Korea, majoring computer science and statistics in 1983 and the MS degree from Hankuk University of Foreign Studies, Republic of Korea, majoring computer science in 1986. After graduation he worked for Agency for Defense Development, Republic of Korea, for six and a half years as a researcher. Having retired from the agency, he has continued his study as a Ph.D. student of the department of computer and information science and engineering, University of Florida. His interests includes data mining or knowledge discovery in databases and related fields of data mining which comprise machine learning, databases, operating systems, and statistical methods.

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Douglas D. Dankel II, Chairman
Associate Professor of Computer and Information Science and Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

LiMin Fu
Associate Professor of Computer and Information Science and Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Sharma Chakravarthy
Associate Professor of Computer and Information Science and Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Paul W. Chun
Professor of Biochemistry and Molecular Biology

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Howard Beck
Associate Professor of Agricultural and Biological Engineering

This dissertation was submitted to the Graduate Faculty of the College of Engineering and to the Graduate School and was accepted as partial fulfillment of the requirements for the degree of Doctor of Philosophy.

December 1998

Winfred M. Phillips
Dean, College of Engineering

M.J. Ohanian
Dean, Graduate School